UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Muhammad Zubair

A Blockchain Solution for Auditing of Timber-to-Charcoal Process

Master's Thesis (30 ECTS)

Supervisors: Mubashar Iqbal
Dr. Fredrik Payman Milani

Tartu 2021

A Blockchain Solution for Auditing of Timber-to-Charcoal Process

Abstract

Blockchain technology is emerging not only in cryptocurrencies like bitcoin and ethereum but also expanding in other business fields. Due to its diverse nature of architecture, it can easily integrate with such environments, e.g., permissioned or permissionless, state-full or stateless. This article mainly focuses on permissioned blockchain, the hyperledger fabric, how to implement it, and how to use it to automate the audit to the timber-to-charcoal business process.

In order to accomplish this study, a design science approach is selected. In this thesis, a timber-to-charcoal case study is analyzed, designed the software architecture, developed the solution using hyperledger fabric, tested and evaluated it with defined evaluation criteria. The technology stack to develop the smart contract and web services to invoke the chaincode and access the ledger is NodeJs.

The final evaluation shows that hyperledger fabric is a secure, fast, and decentralized immutable ledger. In conclusion, it is a suitable approach and benefits the audit to timber-to-charcoal business process.

Keywords:
Blockchain, hyperledger fabric, permissioned blockchain, blockchain solution, design science, timber-to-charcoal

CERCS: P170 (Computer science, numerical analysis, systems, control)

Plokiahela lahendus puidust söe protsessi auditis

Resümee

Plokiahela tehnoloogia ei arene mitte ainult krüptovaluutades nagu bitcoin ja ethereum, vaid laieneb ka muudes ärivaldkondades. Arhitektuuri mitmekesise olemuse tõttu saab see hõlpsasti integreeruda selliste keskkondadega, nt loaga või loata, olekuga või olekuvaba. See artikkel keskendub peamiselt loaga plokiahelale, hüperleederi kangale, selle rakendamisele ja sellele, kuidas seda kasutada puidust söe protsessi auditi automatiseerimiseks.

Uuringu teostamiseks valitakse disainiteaduslik lähenemine, mille käigus analüüsisime juhtumiuuringut, kujundasime tarkvaraarhitektuuri, töötasime välja lahenduse hüperleederkanga abil, testisime ja hindasime seda määratletud hindamiskriteeriumitega. Tehnoloogiapinn nutikate lepingute ja veebiteenuste arendamiseks kettkoodi käivitamiseks ja pearaamatule juurdepääsuks on NodeJs.

Lõplik hindamine näitab, et hüperraamatukangas on turvaline, kiire ja detsentraliseeritud muutumatu pearaamat. Jõudsime järeldusele, et see on sobiv lähenemine ja toob auditile kasu puidust söeks protsessile.

Võtmesõnad:
Plokiahel, hüperleedri kangas, lubatud plokiahel, plokiahela lahendus, disainiteadus, puit söeks

CERCS: P170 (Arvutiteadus, numbriline analüüs, süsteemid, kontroll)

Contents

1 Introduction

Robust audit systems may assist companies in reducing and eliminating a variety of hazards, such as the danger of significant financial statement falsification. It also lowers the risk of asset theft, fraud, and poor management due to inadequate operational knowledge [30]. This study is related to the development of an audit system for the timber-to-charcoal process. The development of audit system is using one of the latest and grooming technology known as blockchain. The hyperledger fabric is one of the variants of blockchain used explicitly for development purposes due to its compatibility with diverse organizational structures, providing security, decentralization, and the ability to cluster different groups of organizations for the separation of concerns to perform their business transactions

1.1 Motivation

The enterprise-level organizations involve multiple businesses and the interaction between those businesses. The significant problem in a multi-business environment is the confidentiality of a particular business context. Every business has its processes and documents, and they need to secure them while maintaining the business transactions with other organizations [4]. For these purposes, many different third-party software solutions provide their services to enable communication between organizations. However, those solutions still have a complex distributed structure and numerous trust issues. One organization agrees with such software's terms and conditions and usage, and most probably others do not.

In order to overcome this problem, the best way is to use a blockchain-based solution considering its built-in nature of decentralization. Blockchain itself is not enough because it allows multiple users to participate without separation of concerns and open access to make the transactions. However, some organizations need to restrict the transactions and do not want to share their information and need some private space. This situation is well formalized and covered in another variant of blockchain technology which is the hyperledger fabric that specifically focuses on enterprise level.

Blockchain also provides encryption and ensures the transaction is not tempered by validating the transaction with the help of all other nodes connected with the network. Hyperledger fabric provides additional features to prevent unauthorized access with the help of MSP. It enables the separation of concerns by creating different channels clustering different nodes to perform relevant operations [1].

Hyperledger fabric can efficiently prevent cheating, faking, and procrastination because of its open, non-tamperable, and traceable nature [17]. in audit processes, the auditor needs to check the validity and quality of data that needs to be consumed while auditing. It is impossible to delete or temper the data in blockchain once it uploads to the chain. However, if it needs to change the data, the approval of 51% of the blockchain participants is required, which is quite a massive ratio if the blockchain network is quite extensive. Also, in traditional audit processes auditor needs a complete balance sheet to perform an audit by requesting it from the organizations and then analyze it containing the risk factor of tempered data and delay in availability of organizational representatives to communicate. However, hyperledger fabric allows the auditor to begin the audit immediately as the transaction is made into blockchain [15].

The hyperledger fabric provides SDK with the most popular languages, e.g., Java, Python, Go, Javascript. The crucial part of the hyperledger fabric is that, the smart contracts can also be written in these languages, providing the developers with ease and not forcing them to learn a new or unfamiliar language to develop the hyperledger fabric [1].

1.2 Research Questions

In order to achieve our research objectives, the following research questions are defined.

RQ1: How to build blockchain-based solution to audit the timber-to-charcoal process?

RQ2: How the blockchain-based solution can help the auditing of timber-to-charcoal process?

The focus of this work is to achieve the defined research questions, but not the least. Following the implementation of the blockchain-based solution, the next step is to analyze the results using pre-defined evaluation criteria defined in section 5.1 in accordance with the audit perspective and features offered by blockchain technology.

1.3 Contributions

A timber-to-charcoal case study is discussed in detail and analyzed to extract the core features as requirements in the form of user stories. Based on these user stories, multiple software engineering artifacts are designed to understand the architecture of software solution, which is further implemented.

This thesis comprises three parts. In the first part, I did literature review about different blockchain technologies and discussed their architectures and workflows. I also searched

for related work done and discussed how the blockchain is helping them. I discussed how they integrated blockchain technology into their business processes and how they are getting benefits using blockchain. By summarizing the result of the literature review, I concluded to work with hyperledger fabric.

I selected the design science approach in order to achieve our research questions. In the second part of the thesis, I started analyzing the timber-to-charcoal case study. Firstly I discussed the case study and the proposed solution to the audit process. I identified the audit's pain points and business needs to the timber-to-charcoal business process and formulated the requirements. Based on these requirements, I started designing and developing the solution and created multiple artifacts that helped us view a clear picture of the software solution. After finishing the development of the blockchain-based solution, I tested it with the help of compiled sample data.

The last part of the thesis is the evaluation of the implemented result. I first defined the evaluation criteria and then started evaluating the results. The evaluation criteria are defined to see the pros and cons of the technology I selected. Based on these results, I concluded my results and wind up the thesis with final remarks and future work. I found that, hyperledger fabric have significant benefits to audit process. It provides decentralizatoin, security, immutability and availability to business process.

The thesis structure consists of multiple sections. Section 2 contains the details about the background study mainly related to blockchain and hyperledger fabric. Section 3 includes an explanation of the methodology used throughout this work to solve the research problems. Section 4 is the necessary part which includes all the analysis, design, and development work done. Section 5 consists of the evaluation criteria and discussion of the results and threats to the validity of the system, and section 7 will be the summarized conclusion of the work followed by the future work.

2 Background

This chapter is mainly concerned with the theoretical overview of related technologies and the related work which has already been performed and can help to proceed with research work.

2.1 Blockchain

Blockchain is a distributed, immutable ledger to record transactions of organizational assets [11]. Blockchain technology was initially introduced by bitcoin cryptocurrency but later expanded to multiple digital economies such as financial businesses, auditing processes, traceability of assets, and privacy-preserving [40], and medical data preservation [14] etc. Blockchain is a well-known technology that is used in the development of cryptocurrencies. It is permissionless technology, so anyone can join and leave it on anytime. Blockchain is a list of transactions that propagates and enhance but never reduce. Each transaction is stored in a node that is linked to its parent node as shown in the figure 1, which is required as a key parameter in proof-of-work consensus algorithm [42].

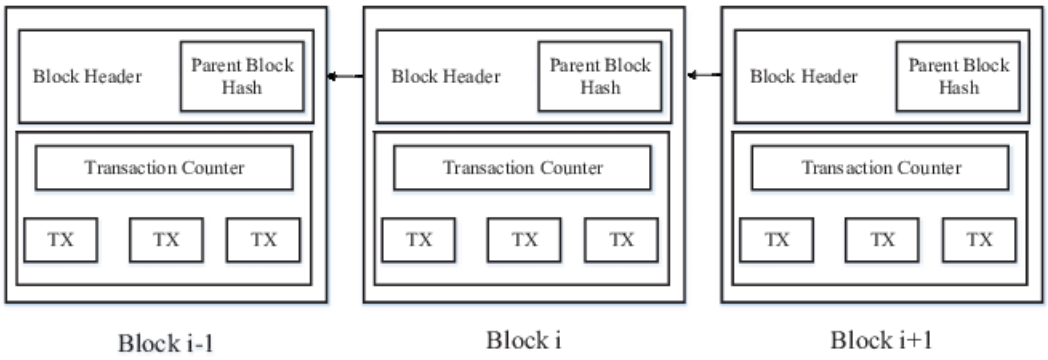

Figure 1. Scheme of Block and Blockchain Structure

The workflow of blockchain is as follows. The user initiates a transaction in the blockchain network. Each transaction represents a block in the blockchain. The block contains the information regarding the transaction, the hash value of the node, the hash value of the previous node, and the timestamp. The block broadcasts throughout all the network nodes. All the nodes validate the node and transaction. The validated block is then added into the chain, and the transaction gets verified and executed.

Each transaction is governed by a digital contract which is known as Smart Contract. The smart contract is an essential component deployed to the blockchain network. It is a

set of rules and logic based on business rules and regulations. Each party that needs to join the network must agree on these contractual rules [35].

2.2 Consensus Mechanism

The consensus mechanism provides the alternative to the trust factor provided by third parties in traditional transaction systems. It helps to validate the transaction before saving the record in the blockchain. Currently, there are three consensus mechanisms widely used listed below [24].

1. Proof-of-Work (POW)
   In this mechanism, miner nodes are assigned a computational task to solve in order to validate the transaction. This mechanism is used in Bitcoin and Ethereum.

2. Proof-of-Stake (POS)
   This mechanism involves the creation of a new block depending on the stake of the validator in the system.

3. Practival Byzantine Fault Tolerance (PBFT)
   This mechanism allows the addition of a new block in the blockchain only when two-third of the validators validates the transaction as successful.

Hyperledger Fabric has its own consensus mechanism, an ordering service intermediate between the sender and receiver. The ordering service is explained in detail in section 2.5. Hyperledger fabric does not implement PBFT but can allow the users to add this mechanism separately according to their needs [29].

2.3 Bitcoin

The very first and successful implementation of blockchain technology is bitcoin cryptocurrency which was first introduced in 2008. It is permissionless technology that helps anyone to join the network without authenticating by anyone. The focus of bitcoin was to eliminate the traditional centralized banking system and provide a decentralized solution without involving any 3rd party agencies. Another concept was to provide anonymity between the receiver and sender of the transaction. The concept of cryptocurrency is to create a payment mechanism based on cryptographic proof rather than trust [22].

Bitcoin technology has its native currency, which is on its own name ***"Bitcoin"***. Each coin is the chain of digital signatures. If the coin owner wants to transfer the coin to another person, he will sign the hash of the previous transaction and the public key of the next person. Then the new block inserts to the end of the chain, and the transaction is successfully done. The transaction verifies with the help of the proof-of-work consensus

algorithm. Once the transaction initiates, The miner node performs the proof-of-work, finds the block hash and broadcasts it to the entire network. In return, the miner nodes earn some bitcoin on every successful block validation. All the nodes in the network validate the hash whether this hash code meets the difficulty level of 18 zeros. If so, this block appends to the chain, and the transaction completes.

Bitcoin solely focused on serving only one thing, that is, to provide the possibility to transfer value from one to another without the need of any central bank. It lacks the transaction of other valuable assets rather than monetary value. On the other hand, to enable the transaction of other valuable assets, blockchain technology has been tailored accordingly and came up with new and improved solutions like ethereum, which is further discussed in the next section.

2.4 Ethereum

Another major blockchain-based technology is ethereum. Similar to bitcoin, it is also permissionless blockchain technology. Ethereum is open-source programmable technology that is not even limited to cryptocurrency or payment but can use for many other digital assets. Bitcoin performs transactions between two peers. On the other hand, Ethereum gives the possibility to perform transactions in a decentralized manner. There are several building blocks of ethereum discussed below in order to understand the workflow [21].

1. Ether (ETH)
   The currency name used in Ethereum is known as Ether.

2. Gas
   There is a certain amount of fee, known as gas, on every transaction depending upon the actual market value of Ether. It helps to defend the network from cyber-attacks such as denial-of-service caused by malicious nodes on the network.

3. Ethereum Nodes

   (a) NVM
   These are particular nodes in the ethereum network responsible for executing the functions written in smart contract. These nodes have minimal access to the network limited to EOA and CA and its storage. These nodes cannot access the overall ledger of the network.

   (b) Mining Nodes
   These nodes write the transaction into the network chain. The miner nodes are awarded by gas whenever they perform the transaction. Only one minor node can write the transaction in the chain. So, all the minor nodes are given

a puzzle to solve, and the first node which solves the puzzle will be the one to write the transaction into the chain.

4. Ethereum Accounts

   (a) Externally Owned Accounts (EOA)
   These accounts are owned by people holding Ether balance in the ethereum network. Whenever a new account creates, a pair of (public/private) keys are generated. The private key stays with the owner safely, and the public key becomes the account identity within the network to perform transactions.

   (b) Contract Accounts (CA)
   These accounts are similar to EOA but lakes private key and hold Smart Contract code.

5. Transactions
   Ethereum transaction is an agreement between two parties in order to exchange assets.

6. Blocks
   Ethereum block is a collection of transactions, and these blocks are connected to form a blockchain, and only the first block is known as the genesis block having no parent node.

7. Smart Contract
   Smart contract is a digitized business rule written in the programming language (Solidity language for Ethereum) mutually agreed by all the account holders to perform business transactions.

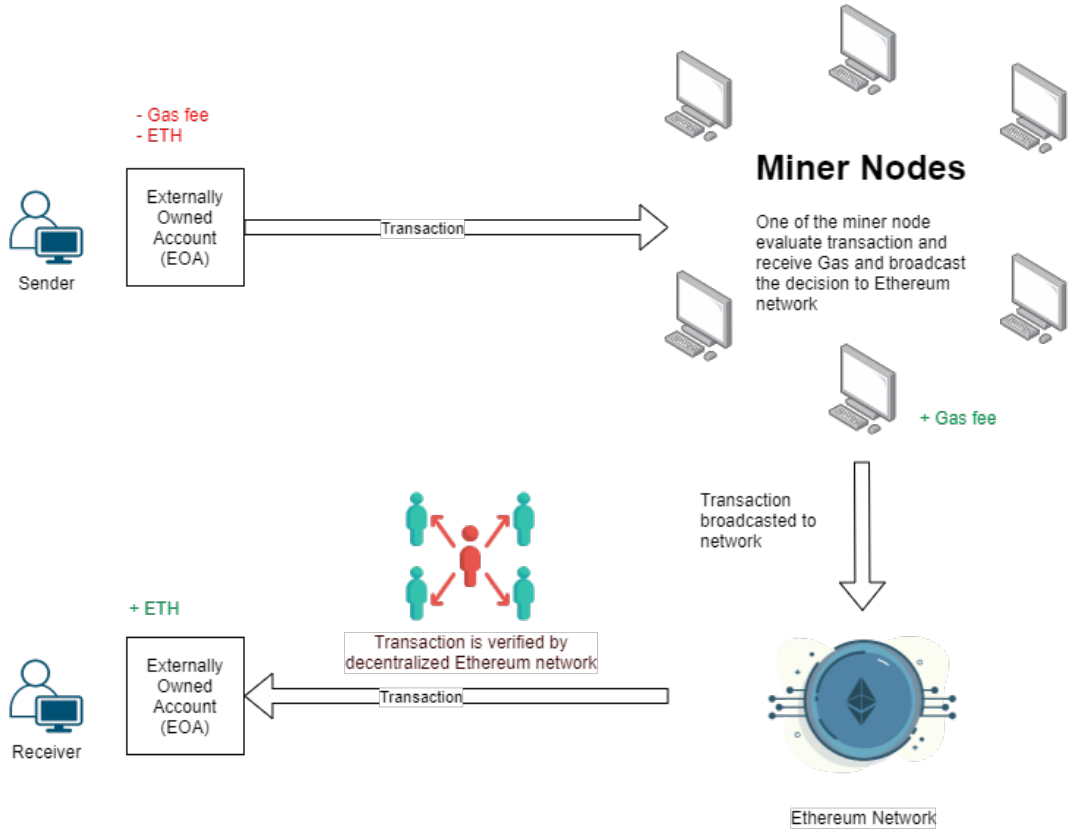

Figure 2. The workflow of etherum to send ether from one user to another

Ethereum transaction begins with the creation of a transaction object containing data. The EOA initiates the transaction by setting up the gas price, gas limit, receiver account address, the Ether amount, and the nonce (transaction count to verify that this particular transaction is not being performed elsewhere). Once the transaction object is created, it is then signed with the private key of the initiator to verify that no one else is performing this transaction on behalf of the initiator. This signed transaction is then submitted to the local node to validate that the initiator account signs this transaction. Once the local node validates the transaction, it then broadcast the transaction object throughout the network and creates a transaction id to get the status of this transaction. The role of miner nodes starts from here to receive the transactions and maintain the transaction pool sorted by gas price. The first miner that evaluates the transaction with the help of the proof-of-work consensus mechanism receives the gas, and the transaction is then added to the blockchain. Then the decision broadcasts to the whole network to prevent evaluating the transaction by other miner nodes. Finally, the receiver gets the value, and the initiator node syncs the transaction with the local copy of the blockchain [37]. The explanation of the workflow of ethereum illustrated in figure 2.

Although, Ethereum provide more programmability and provide more feature as compared to bitcoin, but still not reliable in the sense of scalability and security perspective [41].

2.5 Hyperledger Fabric

The blockchain technology was introduced to support and develop a digital currency, and it is permissionless. However, the bitcoin and ethereum is permissionless technologies, and the speed of transaction is very slow because of proof-of-work consensus mechanism [7]. With the passage of time, this technology was improved and groomed in other professions as well. It was further improved and came up with permissioned technology such as hyperledger fabric [6]. It is distributed and has an access control mechanism to disable unwanted users from accessing the network, which is not present in the bitcoin and etherum. It also has channels and organizations, so if there are multiple organizations and each one of them has its own smart contracts, then they can make their transactions separately for separation-of-concerns without interrupting the other network infrastructure. Therefore, considering these differences I proceed with the option to use hyperledger fabric in our study.

Let's get to the context to discuss hyperledger fabric in deeper level, which is a distributed operating system for permissioned blockchain technology introduced by IBM and hosted by Linux Foundation, which is now broadly being used to automate a different kind of BPMN processes. Fabric is a permissioned distributed blockchain written in (e.g., Go, Node.js and Java). Its architecture is involved an execute-order-validate algorithm which is not the standard flow of blockchain to make it unique and customized to support the business needs [3]. Hyperledger fabric network consists of following components [18]:

1. Assets
   Asset can be every valuable thing e.g. Money, Plant, Bus, Car, or any object which is the part of business assets having some monetarily or business value. Asset have some state which needs to be stored and have an ownership. In terms of hyperledger fabric, Assets stores as a collection of states in Ledger in the form of Key-Value Pairs.

2. Shared Ledger
   Blockchain consists of list of blocks, and hyperledger fabric needs to store these block in the form of states in digital shared ledger. Ledger have two main components: World State and Blockchain.

3. Smart Contract
   In terms of hyperledger fabric the smart contract is know as chaincode which holds

the business logic of the system. Chaincode is responsible to interact with the ledger and perform transactions to the Blockchain.

4. Peer Nodes
   Peer nodes are core components which host the ledger and chaincode. There are three type of Peer Nodes:

   - Endorsing Peer Nodes
     These nodes are responsible to create the transaction proposal in single chaincode container based on the chaincode results. Chaincode installation is compulsory on these nodes.
   - Committing Peer Nodes
     These nodes just hold full ledger and do not invoke any chaincode function.
   - Ordering Peer Nodes
     These are the most special peer nodes. These nodes contain the entire ledger transaction history includes valid and invalid transactions. Other types of nodes only hold a valid transactions history. Moreover, these nodes are responsible for receiving the endorsed transaction proposal and send it to all other nodes to validate that transaction and update their Ledgers.

5. Channel
   It is a logical unit to form a cluster of peer nodes to perform transactions within the channel regardless of letting know other peers who belong to different channels.

6. Organizations
   Organizations are the members of the network who have one or more Peer Nodes within the network. Multiple organizations participate in forming hyperledger fabric in which peers of inter-organization interact with each other and fulfill their business needs.

7. Membership Service Provider (MSP)
   MSP is certificate manager to authenticate the user on the network. It is implemented as a Certificate Authority (CA). Certificate Authority is responsible for the enrollment of users to the network, invokes the blockchain transactions, and ensures the secure connection between the users and the components of the blockchain network.

8. Ordering Service
   Ordering service creates a package of transactions and transfers between different types of peer nodes. Furthermore, It makes sure the transaction is delivered over the network. Only two Solo and Kafka configuration mechanisms can be used for ordering service.

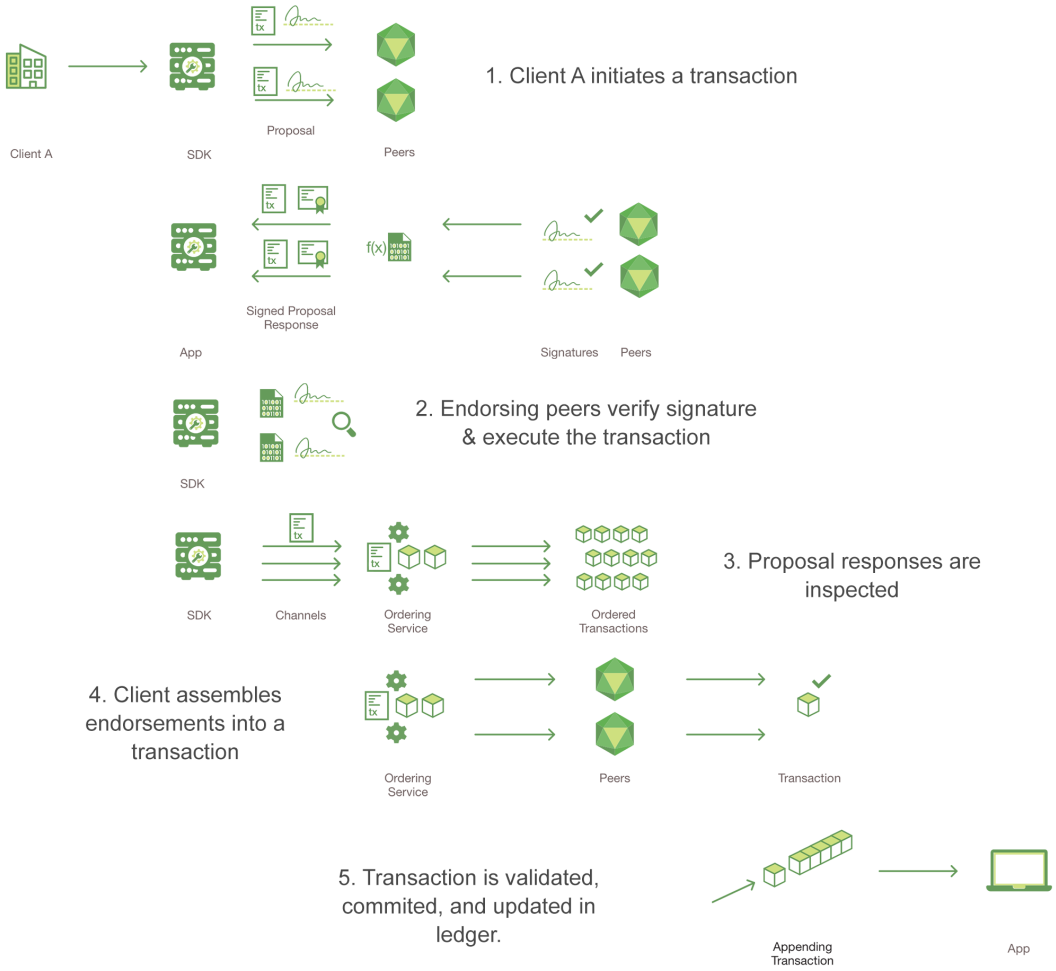

Figure 3. Illustration of a transaction flowing through various components of hyperledger fabric

Blockchain integrates and fulfill business needs without modifying the core business processes. Traditionally, organizations tend to replace old technologies with new ones by giving the cost to reorganize them. Hyperledger fabric is a permissioned blockchain that allows organizations to enhance their business performance because of its capability to behave as ERP systems for such organizations as I am considering the audit to timber-to-charcoal business process.

2.5.1 Why Hyperledger Fabric?

Due to different business needs and the nature of organizational perspectives, it should be clear that which blockchain type makes sense and considerable fulfillment of desired needs. If the transaction state needs to be stored, there are multiple writers to the blockchain, but all the writers are not known, then permissionless blockchain should be used. If all the writers are known but not trusted, then permission blockchain should be used, and if public verification is required, then public permission blockchain should be used; otherwise, private permissioned blockchain should be used. If transaction state is not needed to be stored and there are no multiple writers to the blockchain and multiple writers, but all are trusted, then it is not suitable to use blockchain [38]. The more detailed graphical representation can be seen in figure 4.

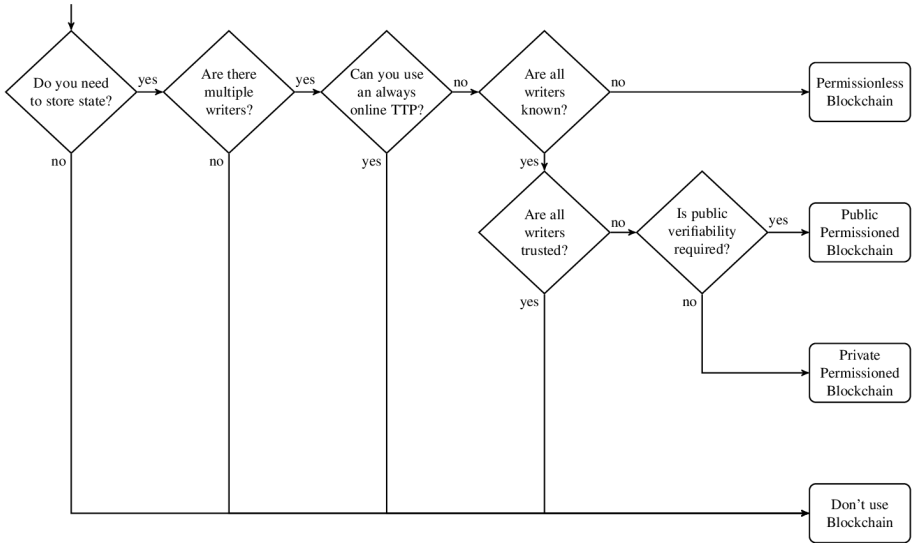

Figure 4. Do you need a Blockchain?

The well-known permissionless blockchain technologies include bitcoin and dthereum, which are open and decentralized, so any user can join as well as leave the network anytime. No legal entities are managing and governing the membership because of its transparent nature. The permissioned blockchain is specifically used when a certain number of users are allowed to read and write into the ledger so that not having all the transaction types remain publicly open to all the users, such technologies are hyperledger fabric, and R3 Corda, etc. [38]. Hyperledger fabric guarantees the security of transactions rather than other blockchain technologies [41] discussed in the previous section.

2.6 Related Work

Hyperledger fabric provides decentralized and immutable solutions to various business fields such as pharmaceuticals, healthcare, financial services, educational training, the internet of things, government, and many other supply chains. All these use cases are business-oriented and require transparency in their business process for authenticity and reliability.

In pharmaceuticals, hyperledger fabric is involved helping drug traceability. It is significantly essential to handle drugs in pharmaceuticals across different manufacturers. There are multiple manufacturers producing drugs, which are then further supplied by multiple suppliers to retailers. The pharmacy's end-users (customers/patients) require drugs that must be safely handled, no fake or low-quality drugs, and drugs produced by authentic manufacturers. The pharma supply chain holds a transaction ledger in which all the history of transactions is maintained, known as MedLedger. It holds proof that every transaction is valid and traceable but hard to manage due to the involvement of several organizations. The immutability and decentralization of hyperledger fabric support medledger tamper-proof and provide transparency to every transaction [32].

In the medical health records system, the storage of patients record is a top priority. There are two significant factors regarding the storage of patients records: privacy and security. Various users access the electronic health record system and need to be private and restricted to operate the system. The patient's records have to circulate between doctors, laboratories, and pharmacies and require decentralization because of the involvement of different organizations. The immutable and distributed ledger offered by the hyperledger fabric provides security. The identity management (MSP) imposes restricted access to the record so that only verified users can access the records [2].

The traditional banking system consists of one central system holding trust, and every customer relies on their policies. The compromise of a single loophole can affect the stakeholders of the entire organization. Hyperledger fabric provides decentralization, enabling the users to hold complete transaction history locally synchronized with all other network nodes with the help of consensus protocol. When someone needs to make a transaction, it will first be validated by all other participants in order to process the transaction. The smart contract behaves as a digital contract between all the participants that limit the human intervention. The computer program is responsible for the approval of the secure transaction, which overrides human errors and personal human intentions in the sense of fraud [5].

Internet of Things (IoT) is the network of inter-connected hardware devices deployed and distributed physically on multiple locations. These devices capture data through sensors

and transmit it to the server to process the data [13]. The primary concern in IoT is data security and decentralization of deployed hardware devices [39]. As the network grows, it is also challenging to manage the devices with one central system. Hyperledger fabric encounters this challenge and provides a solution to connect devices in a decentralized network [16].

The certification or educational document generated by universities or other educational institutions faces the problem of issuing fake degrees and certificates by unknown individuals to gain higher admissions to getting higher jobs. To check the authenticity of the document, educational institutes or employers require the attestation of the document by some trusted facility. Due to a massive load of students every year, these trusted facilities face a problem of overburden and take so much time to process the documents. Using hyperledger fabric, the institute owner will issue the certificate to the certified person and create an identity of each certified person. The Issuer will provide specific access for the sake of confidentiality to the organization that needs to view the authenticity of the certificate of a certified person. It will eliminate the need for trusted third-party attestation facilities and minimize the time to check the authenticity of a document [28].

It is apparent from literature that the hyperledger fabric provides a wide range of benefits to organizations which I discussed in this section. To summarize this section, I can say that the hyperledger fabric facilitates the pharmaceutical industry for better evaluation, payment, and transparency. It provides privacy and removes counterfeit drugs in healthcare industries. The financial industries benefit from accountability, confidentiality, and privacy. It offers a backbone structure to the devices for communicating securely in IoT. It provides transparency and traceability to supply chain items. It eliminates the need for trusted third-party organizations to attest the educational documents such as certificates and degrees offered by educational institutes. These features of hyperledger fabric can help us in building the immutable, tampered-proof, confidential and transparent audit to timber-to-charcoal business process.

3 Research Method

Engineers and scientists widely use the design science research approach to accomplish their innovation, ideas, practices, and products by analyzing, designing, implementing, managing, and using the information effectively and efficiently. Firstly, some research work is required to analyze the existing solutions further to start implementing the project. The best suitable methodology suggested according to the scenario is design science methodology [10].

Design science methodology is specifically related to developing the innovative solution and mainly focuses on the best possible way to create the solution rather than what currently exists as mentioned in Guideline #2 [10]. This approach aims to provide knowledge to the related professionals to reuse it in their field problems.

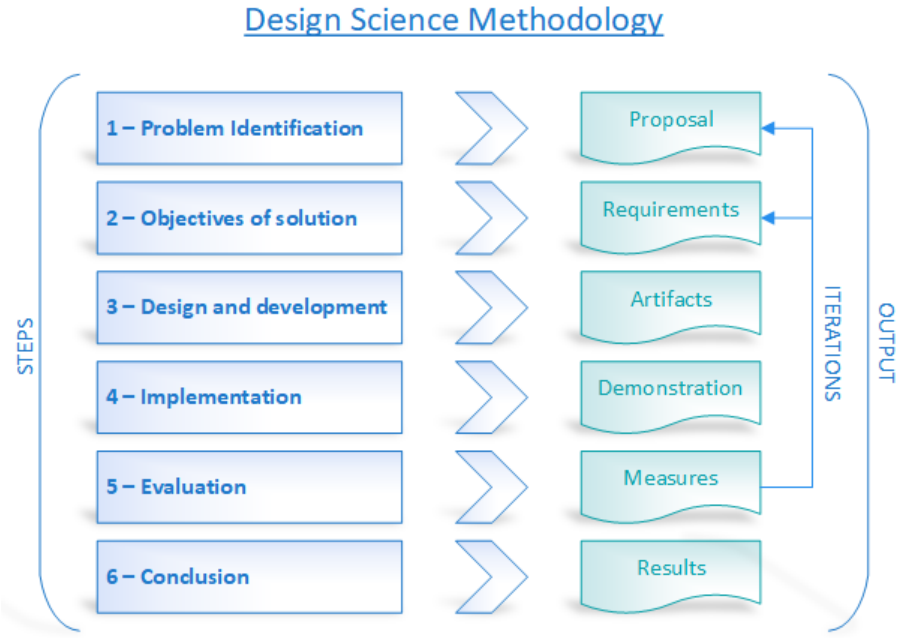

Figure 5. Design Science Methodology - Work Flow

The design science approach comprises six steps. The illustration of workflow followed by this approach is present in figure 5 [25]. The process begins with the problem identification step, where it is required to do research work about the problem and search for existing available solutions. Then identify the main problem and propose the possible solution to the problem. Right after giving the proposal, the next step is to define the objectives and goals of the solution. Usually, this step is to define the scope of the solution

and formalize the requirements to understand the exact need that needs to be developed. When scope is defined and requirements finalized, the designing and development step initiates. This step includes requirements engineering processes.

Different requirements engineering activities are performed to analyze the requirements and transform those requirements into different artifacts valuable for the technical teams such as developers and analysts to understand the system in detail in the context of implementation.

1. Problem Identification
   The problem identification step includes understanding the target case study discussed in section 4.1 timber-to-charcoal process. Some other research works have deeply analyzed this process, and they proposed possible solutions. The section 4.2 refers to their proposed solutions, which are summarised here to extract the core requirement. This study extends those proposed solutions to design, develop and provide a working prototype of their proposed solutions.

2. Objectives of Solution
   After defining the proposed solution, the next step is to formulate the requirements. All basic requirements extracted from the case study are written in User stories mentioned in section 4.3.2 User Stories.

3. Design and Development
   Section 4.3 Solution Design refers to this step. Different analysis techniques were applied to design the software architecture to understand what the system needs to build and how different users will interact.

4. Implementation
   Section 4.4 Implementation belongs to this step. After analyzing and designing different design artifacts, the development started, implemented, and discussed each aspect in detail.

5. Evaluation
   The design science approach suggests multiple evaluation methods: Observations, Analytical, Experimental, Testing the solution, and through description method [10]. I followed Experimental and Testing evaluation methods. The experimental method aims to validate the solution in a closed and controlled environment, as I will do to test the immutability of hyperledger by setting up a test environment.

   I evaluated the solution with the help of defined criteria described in section 5 Evaluation. Defining the criteria in section 5.1, discussion about the results according to the evaluation criteria in section 5.2.

6. Conclusion
   Design science methodology supports multiple iterations to redesign the solution based on measures taken by evaluation. This study considered only one iteration and concluding its results in section 7 Conclusion.

Our study should meet three objectives to consider effective research as mentioned in the article [25]. The first objective is, our study should be consistent with the previous literature. I performed the literature review and remained consistent with the blockchain-based implementation and proposed solution. The second objective is, it should provide a nominal process model on which I conduct our research. I accomplish this objective by creating multiple design artifacts that involve use cases, sequence diagrams, ERD, and class diagrams. The third objective is to present a model on which I perform the evaluation. I accomplished this objective by developing a hyperledger fabric network, and then I evaluated it with the help of defined evaluation criteria leading to the conclusion of our study.

4 Case Study

This chapter focuses on the process and product realization of audit to the timber-to-charcoal business process. This chapter will explain the business process, design the software architecture, develop the audit solution, and test the outcome of the developed solution. This chapter also provides the answer to our research question (RQ1). The section 4.1 and section 4.2 refers to the problem identification step, which is very first step of our selected research method, as shown in figure 5.

4.1 Timber-To-Charcoal Business Process

Let's discuss the timber-to-charcoal business process. The certification and auditing of timber-to-charcoal process are considered as target scenario cases. In general, the process starts from cutting timber wood by owners then sending wood to charcoal processors directly or by some brokers. The processors process the wood to make charcoal at the conversion rate of 80%, which means only the 20% of entire input remains as output. Then the processed Charcoal is sent to the secondary processors who transform coal into packages on the conversion rate of 90% means 10% volume wasted during packages. Then finally, packed charcoal sells to retailers directly or by the brokers. Certification and Audition are performed by a third-party audit company. A basic structure of the process is shown in figure 6.

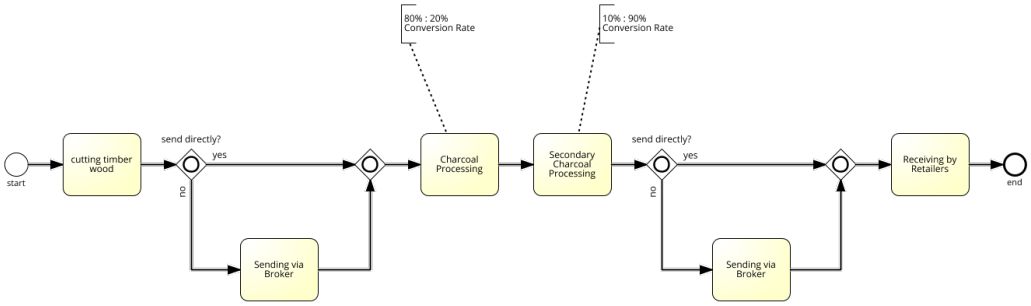

Figure 6. A basic flow of timber-to-charcoal business process

Certification and auditing take place in two steps. Firstly, the initial certification and the second step is yearly auditing which is continuous and repetitive. The process starts with signing the contract, and then the initial inspection begins with the checking of environmental conditions, then documentation checking, and supplier's certificates. Sometimes inspector checks the original invoices if it required for in-depth evaluations. At the end of the audit, the report is checked by another inspection officer who takes a decision. After getting the positive decision, the certificate is issued to the company

and recorded in the FSC database. After passing a year, the second step of certification begins. The second step is the same as the first except signing the contract and issuing the certificate.

4.2 Proposed Blockchain Solution

Previously, the timber-to-charcoal process is taken as a case study in one of the research work and analyzed very deeply in terms of business process management [19]. The process is modeled as-is then analyzed in detail to find the non-conformance in the auditing process. The author mentioned five non-conformance points which are mentioned below.

1. Planned Audit time
2. Compiling audit report with random numbers
3. Audit takes place once a year
4. Supplier's certificates checked against FSC database which is also updated once a year
5. Inspectors time and interest

The author proposed and presented the blockchain-based solution in order to solve the non-conformance to fulfill the business needs, which is shown in the figure 7.

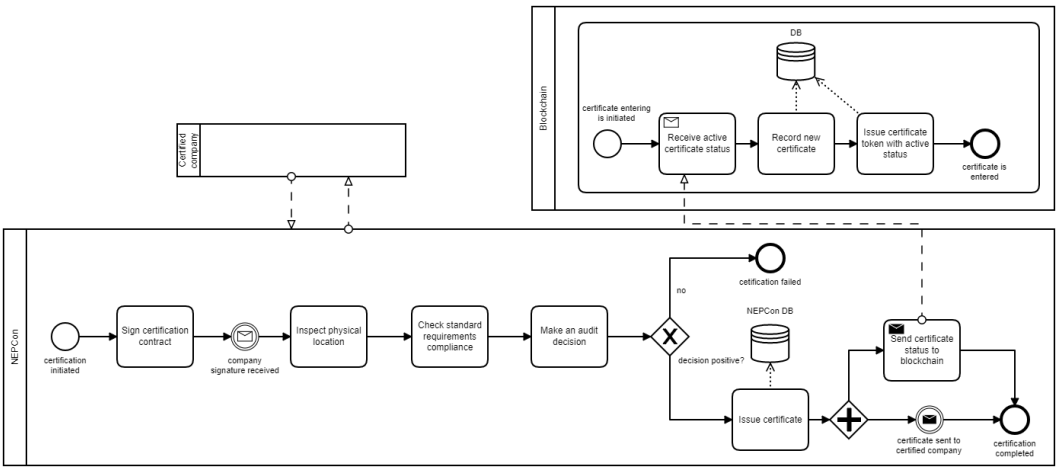

Figure 7. A proposed blockchain based solution to timber-to-charcoal business process

However, the result of this work is very high level in terms of development. There is another research work concluded in which a more detailed solution is proposed. The

authors used in-depth analysis to present the more validated and reliable BPMN model of auditing the timber-to-charcoal blockchain-based solution as you can see in figure 8.

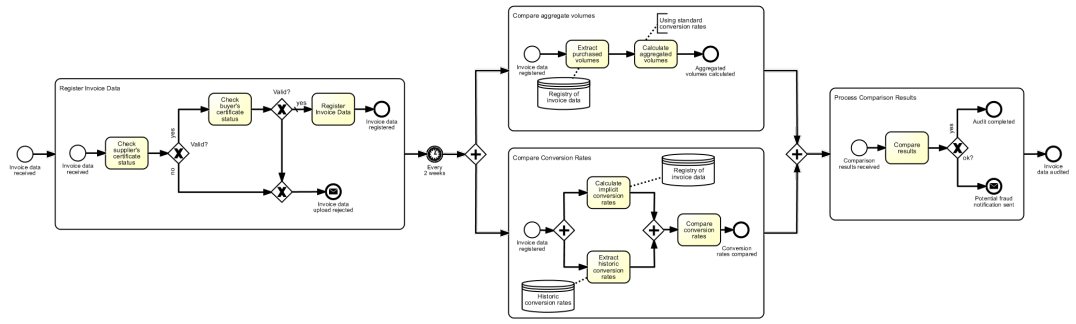

Figure 8. A detailed design of audit process in terms of smart contract

The process begins with the registration of invoices. Invoice data will only include *Invoice Id*, *Product Id*, *Volume*, *Seller*, *Buyer*, *Date*, and *Invoice Hash*. The hash value is just for the integrity of invoice data to validate that the provided invoice data is not tempered or somehow unintentionally changed. When the system receives Invoice data, first, it will check the certificate status of the supplier. If the status is valid, it will proceed for further checks; otherwise, invoice registration will be canceled. If the status is valid, then the buyer's certificate status will be checked. If the buyer's certificate status is also valid, then the invoice data will be successfully uploaded to the blockchain; otherwise, it will be canceled.

The second phase in the process is to perform an audit. The audit process will start after the two weeks are finished because there will not be enough invoices, and the audit result will not be as accurate if the audit starts earlier. The audit process consists of three parts, *Compare Aggregate Volumes*, *Compare Conversion Rates*, and *Process Comparison Results*.

The *Compare Aggregate Volumes* will extract the invoice data, check how much volume the seller purchased, and calculate the percentage of the volume that he sold to the buyer. Then conversion rate will be compared with the standard conversion rate of the buyer. Suppose the volume sold is greater than the volume purchased by the seller, and the percentage of volume sold exceeds the particular threshold compared to the seller's standard conversion rate. In that case, this will be considered as a potential mis-compliance.

The *Compare Conversion Rate* will check the implicit conversion rate of the seller of invoice by calculating with the help of percentage bought and percentage sold. After that, the process will find out the *Historic Conversion Rate* of the seller. Suppose the *Implicit*

Conversion rate deviates the particular threshold comparing the *Historic Conversion Rate*, then this will also be considered a mis-compliance.

Once the result of *Compare Aggregate Volume* and *Compare Conversion Rate* will calculate, the *Process Comparison Result* will then compare both outputs. If any of the processes show mis-compliance, the seller will consider fraudulent behavior, and the relevant certifier will be notified about the mis-compliance [20].

If the certified companies need to update their standard conversion rate, they will not update without requesting a relevant certifier. The certifier will check the request and decide whether the request should be approved or rejected. If the requested change in conversion rate deviates a certain threshold comparing the previous conversion rate, then the request should not be entertained and must be rejected [19].

To summarize the first step of our research method I came up with the proposal to develop a blockchain-based solution to audit the timber-to-charcoal process. This work will be considered my base requirement source for my research work which will be discussed through this article.

4.3 Solution Design

This section belongs to the second and third steps of our research method that is to define the objective of the solution and design the solution in terms of requirements and artifacts, respectively. The solution design is essential according to the third objective of the design science approach. I will produce artifacts in this section that will relate to the business needs of audit to timber-to-charcoal and help produce the working software modal. This section will focus on several inputs which will influence the development of blockchain-based software solution. However, this section will cover the analysis and design for core processes explained in figure 8, excluding the registration of *Certified Companies* and *Certifiers* but still developed and implemented.

4.3.1 Design Artifacts

The design artifact is a construct, model, or method applied in the development of a software solution [10]. These artifacts provide a vocabulary or symbols that help to realize the problem and solutions. The list of artifacts I selected to produce, to identify the problems and solutions to these problems, in audit to timber-to-charcoal business process is listed in table 1.

Artifact	Importance and its purpose
List of user stories	User stories will help to identify the core business requirements. One user story contains the responsible person who need to do some action and specify the reason why he need to perform the specific action.
Use cases	It helps to define the scope of the project by creating the system boundary and actors who involved. With the help of user stories, we define features and functions of the system and the flow of actions performed by the actors. This gives the more precise information about the behaviour of developed system
Entity Relationship Diagram (ERD)	It helps to define the data models of the business context. I can create the data models using this diagram to develop chaincode in hyperledger fabric. Inside the chaincode, these data models will represent an object of transaction data that will be further stored in the ledger.
Sequence diagrams	A sequence diagram will help us to understand the insights of the software solution. In these diagrams, we can see the internal flow of every single function, how these functions perform tasks, and how they create or destroy the data models. It helps to see what kind of exception can occur and what alternative response should generate. This diagram is technical in terms of software development.
Class Diagram	The class diagram is the advanced form of ERD. It helps to identify that how many classes we require to create while developing the solution. How these classes interact with each other, what parameters are required by their member functions, and what access level we have to define for the member functions and attributes of the class. The developer can start writing code with the help of this diagram.

Table 1. The importance and purpose of design artifacts that I selected for the solution design

4.3.2 User Stories

All the core requirements extracted from the case study have been written in the form of user stories. User stories will help identify the features and functionality, the system response, and the users responsible for performing those functionalities. User stories are enlisted in table 2.

S/n	User Stories
US01	As a Certified Company, I want to upload the Invoice data, so that the data can be used for audit purposes.
US02	As a Certified Company, I want to read the Invoice data, so that I can view the uploaded invoice.
US03	As a Certified Company, I want to create a request to change the conversion rate, so that Certifier notified about the request.
US04	As a Certifier, I want to register the Company, so that they can upload invoices.
US05	As a Certifier, I want to register new Certifier, so that they can initiate an audit.
US06	As a Certifier, I want to perform Audit, so that fraudulent behaviour of companies can be identified and notified to relevant Certifier.
US07	As a Certifier, I want to see the conversion rate change requests, so that I approve or reject it.
US08	As a Certifier, I want to change the company status, on the bases of audit result.

Table 2. User stories defining the system core requirements

User stories only explain the features on a broader level and do not contain preconditions, postconditions, and exceptions. Use cases will explain these details, which are discussed in section 4.3.3.

4.3.3 Use Case

One of the best ways to visualize the user stories graphically with broader views is to represent and transform the needs mentioned in user stories into use cases and use case diagrams, as shown in figure 9. Each oval-shaped use case has its detailed technical functionality, which has been discussed individually.

There are two main actors, i.e., *Certified Companies* and *Certifiers*, drawn on the left side of the box. The *Certified Companies* can upload invoice data, read invoice data, and create a request to notify the *Certifier* to change the conversion rate. The *Certifiers* have more access to the functionality of the system. The *Certifiers* can read a complete list of invoices and the history of each invoice. The *Certifiers* can read the notifications and have access to resolve them based on their decisions.The *Certifiers* can also register, read, and change the company status. The most complex and crucial use case to perform audit which is also only accessible by *Certifiers*. Perform audit use case is extended by process comparison result, further extended by two use cases, i.e., *Compare Conversion Rates* and *Compare Aggregate Volumes*.

Figure 9. The scope of the system, in the form of usecases and actors who interact with the system

4.3.3.1 Use Case #1 - Upload Invoice

In this use case, the certified companies as users will upload the invoice data to the system. Invoice data will save into the ledger only if the buyer and seller data are ACTIVE; otherwise, the transaction will be canceled.

UC01	Upload Invoice	
Dependencies	User story number US01	
Description	The system shall behave according to the described use case when Actor try to upload the invoice data.	
Pre-condition	Certified company must be registered, and have valid certification status in order to upload the invoice data.	
Ordinary Sequence	Step	Action
	1	Actor Certified Company provide invoice data to the system.
	2	System checks the certification status of seller and buyer.
	3	System informs that the invoice data has be uploaded.
Post-condition	System has responded about the data uploaded.	
Exceptions	2	Use case will be cancelled if the status of buyer and seller is suspended.
Comments		

Figure 10. A use case for uploading the invoice to the system

4.3.3.2 Use Case #2 - Create Notification

In this use case, the certified companies will request to change the conversion rate, and the relative certifier will be notified about this change request.

UC02	Create Notification	
Dependencies	User story number US03	
Description	The system shall behave according to the described use case when Actor makes a request to change the conversion rate.	
Pre-condition	Certified company must be registered, and have valid certification status in order to upload the invoice data.	
Ordinary Sequence	Step	Action
	1	Actor Certified Company provides new conversion rate which needs to be changed with old one, along with notification type.
	2	System validates the type of notification and data provided by the Actor and creates a notification.
	3	System informs to the Certifier about the notification.
Post-condition	System notified the relevant certifier about the conversion rate change request.	
Exceptions	2	Use case will be cancelled if the data validation failed.
Comments	If the notification type is Conversion Rate Change, then Actor must have to provide the new conversion rate otherwise use case will be cancelled and request will not proceed.	

Figure 11. A use case for creating a notification regarding updating exchange rate

4.3.3.3 Use Case #3 - Resolve Notification

In use case UC03A, the certifier will decline the request for changing the conversion rate generated by the certified company. No change will be made in the certified company's data and notification status change to DECLINED.

UC03A	Resolve Notification	
Dependencies	User story number US07	
Description	The system shall behave according to the described use case when Actor resolves the notification by declining it.	
Pre-condition	Certifier must be registered, and fetched the list of Notifications.	
Ordinary Sequence	Step	Action
	1	Actor Certifier provides status which needs to be changed by system.
	2	System validates the data provided by the Actor and the Status of the Notification as Declined.
	3	System informs to the Certifier about the resolved notification.
Post-condition	Status of the notification gets Declined.	
Exceptions	2	Use case will be cancelled if Actor do not provide valid status.
Comments	System will accept only two types of status, i.e., APPROVED and DECLINED. This use case belongs to the notification which have to be declined. SEE use case UC02B if the Actor wants to approve.	

Figure 12. A use case for resolving the notification to decline

In use case UC03B, the certifier will approve the request for changing the conversion rate generated by the certified company. There are two types of notifications; the first one is to change the company's conversion rate. If the certified company approves this notification, then the provided conversion rate replaces the old conversion rate, and the conversion rate of the certified company will be changed. The system generates the second type of notification during the audit process. If the system detects fraudulent behavior in invoice data, the notification will generate and circulate to the related certifier company. On approval of this notification type, the status of the certified company changes from ACTIVE to SUSPENDED.

UC03B	Resolve Notification	
Dependencies	User story number US07 (change conversion rate), User story number US08 (change company status)	
Description	The system shall behave according to the described use case when Actor resolves the notification by approving it.	
Pre-condition	Certifier must be registered, and fetched the list of Notifications.	
Ordinary Sequence	Step	Action
	1	Actor Certifier provides status which needs to be changed by system.
	2	System validates the data provided by the Actor and the Status of the Notification as Approved.
	3	System identifies the type of notification as conversion rate change and change the old conversion rate with new provided conversion rate
	4	System informs to the Certifier about the resolved notification.
Alternate Sequence	Step	Action
	3.a	System identifies the type of notification as fraudulent behavior and change the certified company status to suspended.
Post-condition	Status of the notification gets approved.	
Exceptions	2	Use case will be cancelled if Actor do not provide valid status.
Comments	System will accept only two types of status, i.e., APPROVED and DECLINED. This use case belongs to the notification which have to be approved.	

Figure 13. A use case for resolving the notification to approve

4.3.3.4 Use Case #4 - Change Company Status

In this use case, the certifier company as an actor changes the status of the certified company from ACTIVE to SUSPENDED and vice-versa.

UC04	Change Company Status	
Dependencies	User story number US08 (change company status)	
Description	The system shall behave according to the described use case when Actor wants to change the status of company.	
Pre-condition	Certifier must be registered, and fetched the list of Notifications.	
Ordinary Sequence	Step	Action
	1	Actor Certifier provides status which needs to be changed by system.
	2	System validates the data provided by the Actor and the Status of the company.
	3	System change the company status and informs to the Certifier.
Post-condition	Status of the company gets changed.	
Exceptions	2	Use case will be cancelled if Actor do not provide valid status.
Comments	System will accept only two types of company statuses, i.e., ACTIVE and SUSPENDED.	

Figure 14. A use case for changing the company status

4.3.3.5 Use Case #5 - Perform Audit

In this use case, the certifier company as an actor initiates a request to perform the audit of all the invoices uploaded by the certified companies. The system will check uploaded invoices, find the fraudulent behaviors, and generate a notification to the certified company's certifier. **Note**: the hyperledger fabric does not provide time-based events or triggers so that the audit process can not be initiated automatically after a certain period.

UC05	Perform Audit	
Dependencies	User story number US06 (perform audit)	
Description	The system shall behave according to the described use case when Actor wants to perform audit of the invoices.	
Pre-condition	Certifier must be registered, and time span of at least two weeks have passed.	
Ordinary Sequence	Step	Action
	1	Actor Certifier initiate a request to perform audit in the system.
	2	System collects all the invoiced with in the time span of two weeks.
	3	System calculates aggregate volumes and compare conversion rates of the invoices made by the seller.
	4	System compares the results found in step 2 and step 3 and identifies the fraudulent behavior of company.
	5	System notifies the relevant certifier about the fraudulent behavior
	6	Actor receives a list of invoices with fraudulent behavior identified.
Post-condition	Audit has been performed and relevant certifier has been notified by the system with the fraudulent behavior.	
Exceptions	2	Use case will be cancelled if invoice data is not present.
Comments		

Figure 15. A use case for initiating the audit process

4.3.4 Entity Relationship Diagram

Entity Relationship Diagram (ERD) helps to identify different classes and the relationship between them. It will also help identify the data attributes required to store as a state in Hyperledger Fabric.

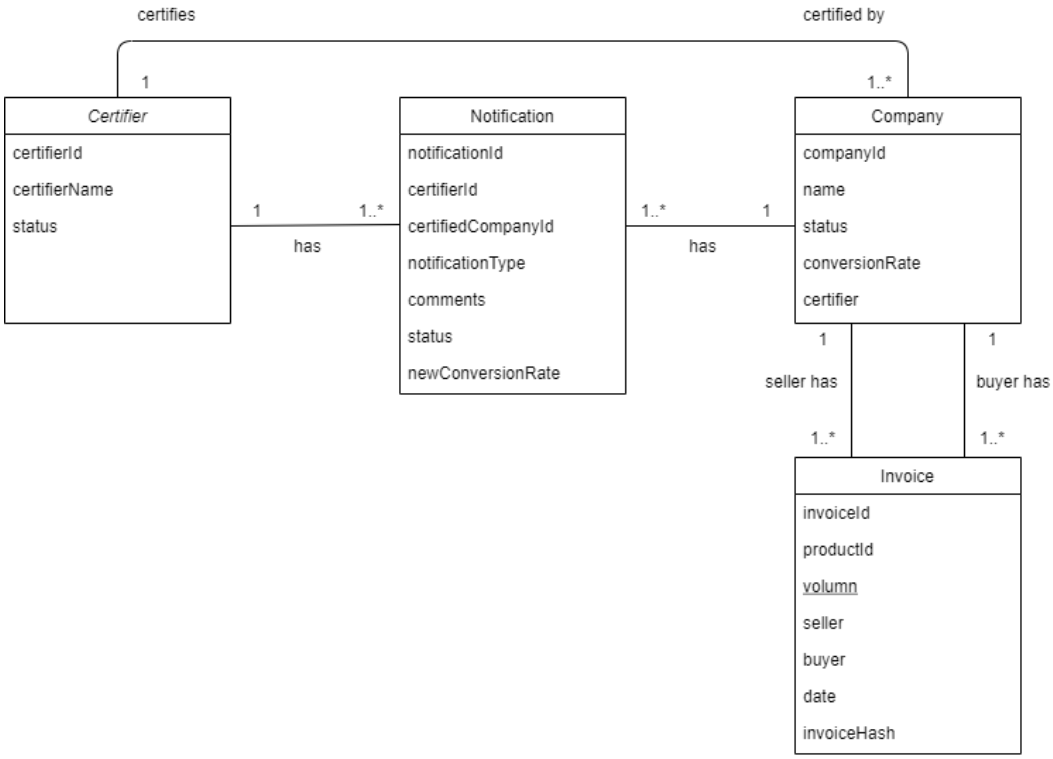

Figure 16. An ERD diagram explaining system entities and the relationship between them

After analyzing the user stories and the case study, four entities were found and modeled in ERD, as shown in figure 16. The *Certifier* entity has attributes, i.e., certifierId, certifierName, and status. The *Certifier* status can be ACTIVE or INACTIVE. The *Certifier* and Company's relationship is that *Certifier* can certify one or more than one *Company* and *Company* can be certified only by one *Certifier*. The *Company* entity has attributes, i.e., companyId, name, status (ACTIVE, SUSPENDED), conversionRate, and certifier. The status of the *Company* can be ACTIVE or SUSPENDED based on audit decisions made by *Certifier*. Another entity is Invoice having attributes, invoiceId, productId, volumn, seller, buyer, date, and invoiceHash. The seller and buyer belong to the *Company* entity, and both can create multiple invoices. However, only one seller can create one invoice, which will belong to only one buyer.

The *Notification* entity can be generated either by the *Company* or can be system-generated. There are two types of notification; one is FRAUDULENTBEHAVIOUR system-generated, and the second is CONVERSIONRATECHANGE which the *Company* creates. The *Company* will create a Notification in the form of a request to change the conversion rate that will further approve or decline by the associated *Certifier*. The

system will generate FRAUDULENTBEHAVIOUR *Notification*, when there is a potential mis-compliance found during the audit process. The attributes of *Notification* entity are notificationId, certifierId, certifiedCompanyId, notificationType (FRAUDULENTBEHAVIOUR, CONVERSIONRATECHANGE), comments, status (APPROVED, DECLINED), newConversionRate.

The hyperledger fabric does not support relational-database. It stores data in the form of states using CouchDB or LevelDB [1]. This ERD has been created to understand and visualize the entities and their attributes in order to conceptualize how we can create the structure of states to store the data.

4.3.5 Sequence Diagram

Each use case has specific steps in order to complete the functionality. Use cases only explain the functionality in generic terms and lack technical information. Each step in the use case occurs as an action. When classes generate, they interact with each other with a specific sequence, which is the exact representation of the action mentioned in the use case. A sequence diagram shows how objects communicate inside the system, giving more technical information about the system's behavior and flow. Sequence Diagrams help to build up the class diagram by enhancing the ERD. Once the class diagram is finalized, It gives the developer more technical information about the system which needs to be developed.

The sequence diagrams explained in this section are just the sequence flow of the chaincode transaction within one peer. The mechanism of hyperledger fabric transaction is discussed separately as shown in figure 17 [12]. In hyperledger fabric possible transactions, a client application connects to one peer node. The application invokes a proposal to communicate with the network in order to perform the transaction. The peer node then invokes the relevant chaincode along with the provided proposal request. According to the proposal, the chaincode is responsible for generating a query to the ledger and updating the response. The ledger submits the response to the related peer node, which then forwards back the response to the client application. Once the transaction is successful, the client application requests the orderer to circulate the transaction to all other peer nodes within the network.The orderer sends the transaction to blocks. Each peer node within the network then updates its ledger and notifies the application about the updated ledger event.

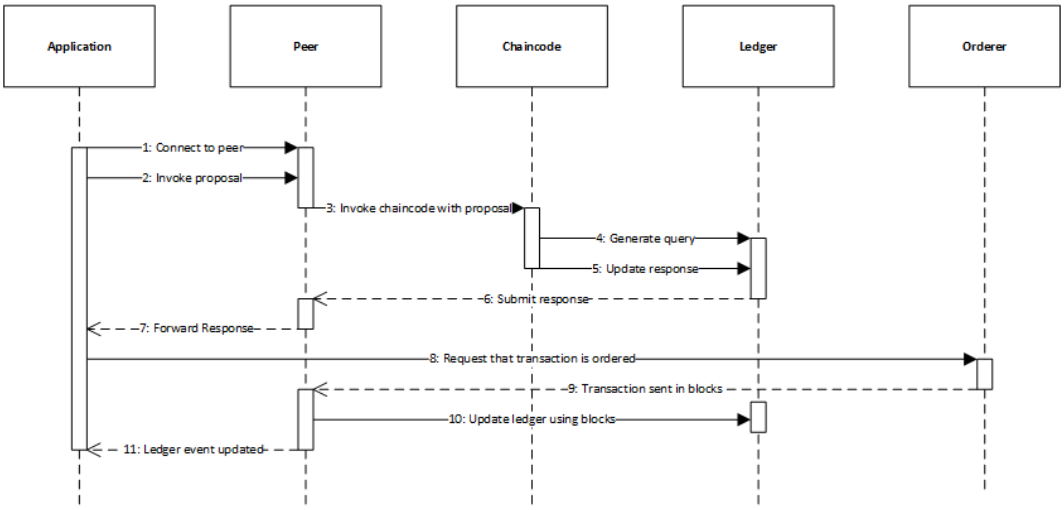

Figure 17. A basic transaction flow of hyperledger fabric

4.3.5.1 Upload Invoice Sequence

The sequence flow of uploading the invoice data is translated from the use case no. UC01 mentioned in section 4.3.2.1. The transaction begins when the actor *Certified Company* submits the invoice data using the web service layer that connects to the fabric network. Suppose the connection gets successful and the user is authorized to perform the transaction. In that case, the web service layer initiates a proposal request using "submitTransaction()" method holding the parameters, the chaincode method name, and the data. The chaincode receives the invoice data and begins further processing. The chaincode gets the seller and buyer details from the ledger and check the status if both the company have ACTIVE status, then the invoice will save into the ledger. Otherwise, the transaction will be canceled, and the invoice will not upload to the ledger. The sequence diagram of this process is shown in the figure 18.

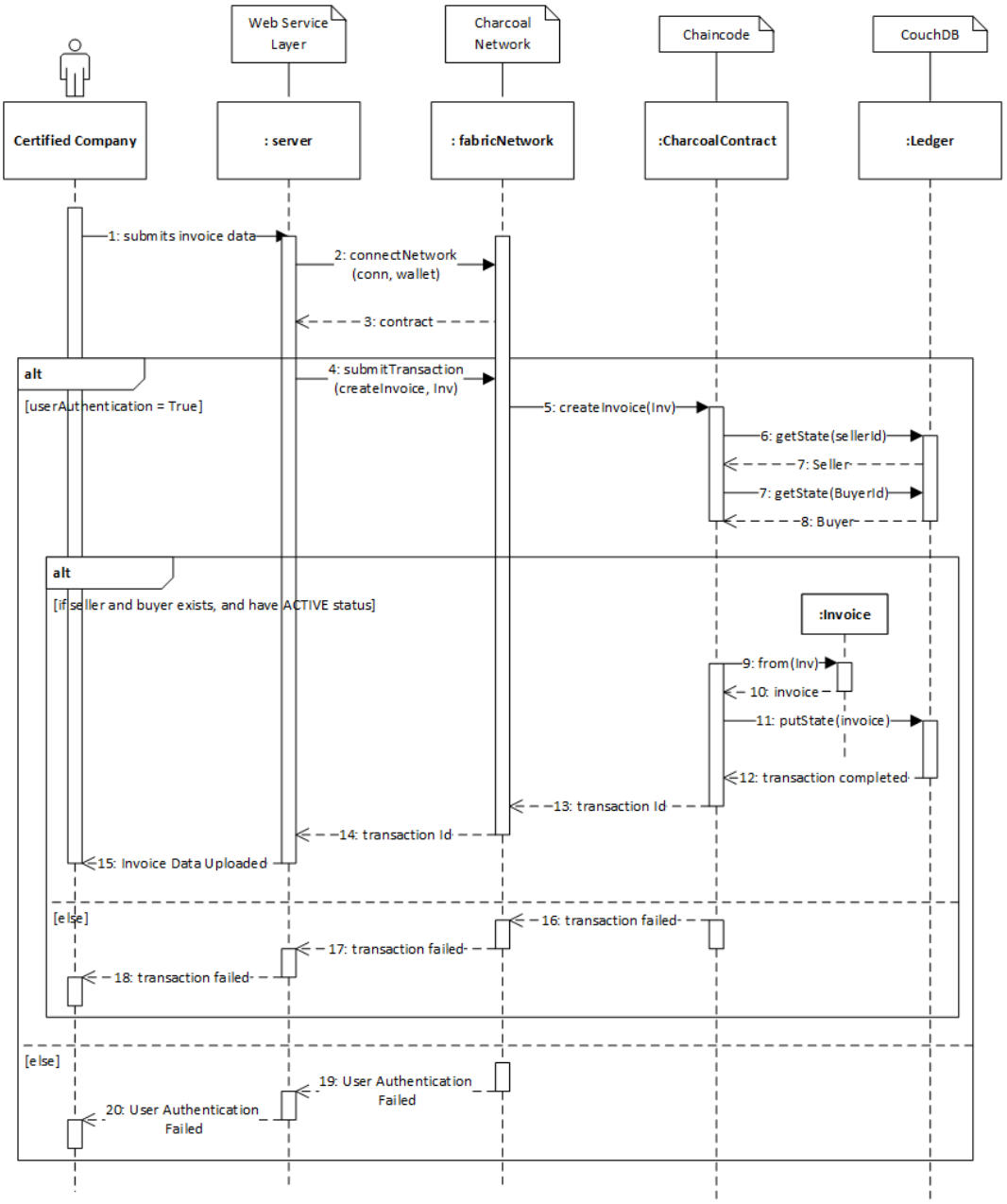

Figure 18. A sequence of uploading invoice to the system

4.3.5.2 Create Notification Sequence

The sequence of authorizing the user and connecting to the network is the same throughout all the diagrams. The *Certified Company* initiates a request to change his conversion rate providing the new conversion rate, which needs to be changed. The "createNotifica-

tion()" method of chaincode is responsible for handling this request. The chaincode will process it and save the state into the ledger and notify the user about the transaction. The sequence diagram of this process is shown in the figure 19.

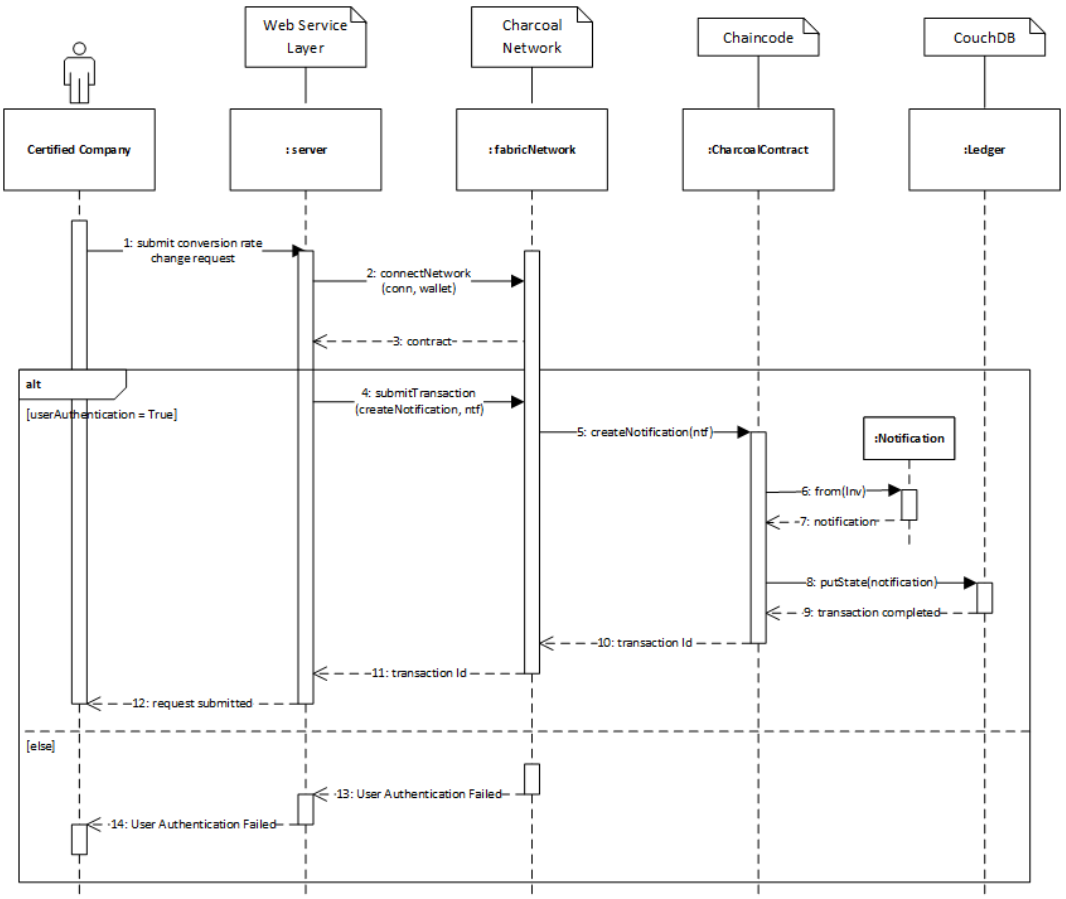

Figure 19. A sequence of creating the notification

4.3.5.3 Resolve Notification Sequence

Only the *Certifier* is responsible for resolving any notification. The *Certifier* can either approve or decline the generated notification. Suppose the *Certifier* declines the notification, chaincode will only update the status of notification to DECLINED. The notification is of two types; one is FRAUDULENTBEHAVIOUR generated by the system while performing the audit, and the second one is CONVERSIONRATECHANGE which *Certified Companies* generate to change their conversion rate. In case of approval of the notification, if the notification type is FRAUDULENTBEHAVIOUR the chaincode will update the related company status to SUSPENDED. If the notification type is CONVER-

SIONRATECHANGE, then chaincode will update the conversion rate with the provided value and save the new state into the ledger. The chaincode will mark the notification status as APPROVED after completing the transaction proposal request. The sequence diagram of this process is shown in the figure 20.

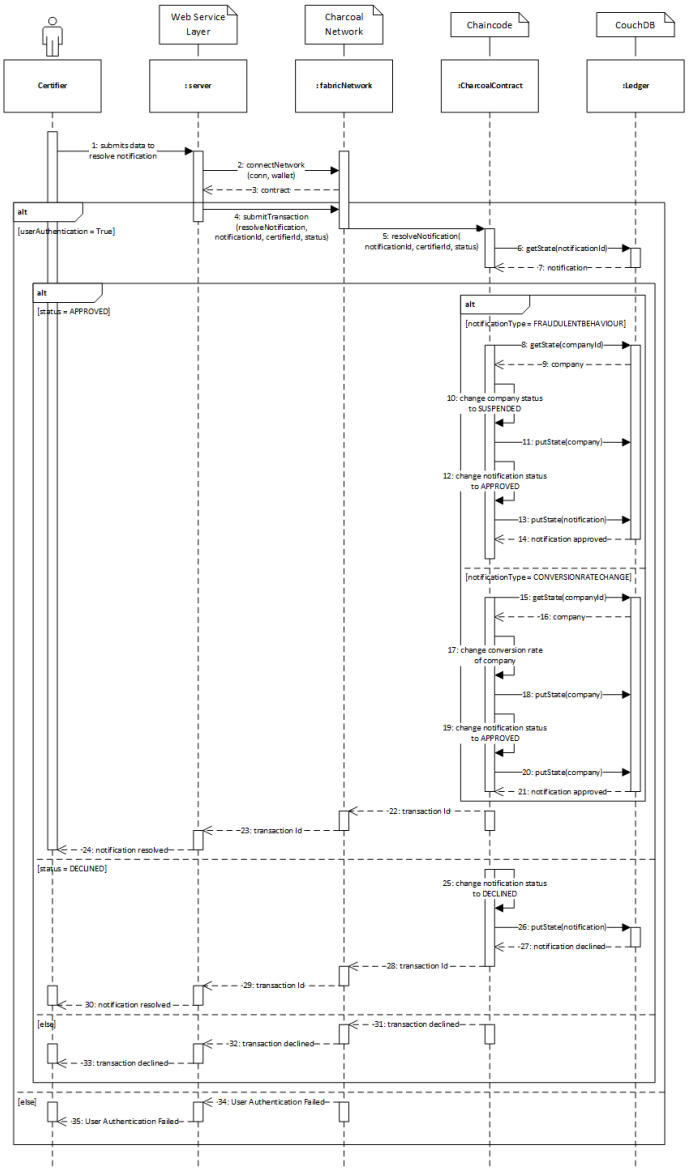

Figure 20. A sequence to resolve the notification as declined or approved

4.3.5.4 Change Company Status Sequence

The sequence of changing the status of company is also simple. The responsible actor for this process is *Certifier*, which will send the company identity and status that needs to be updated. The chaincode will get the company state and update the company's status and submit the updated company details to the ledger. The sequence diagram of this process is shown in the figure 21.

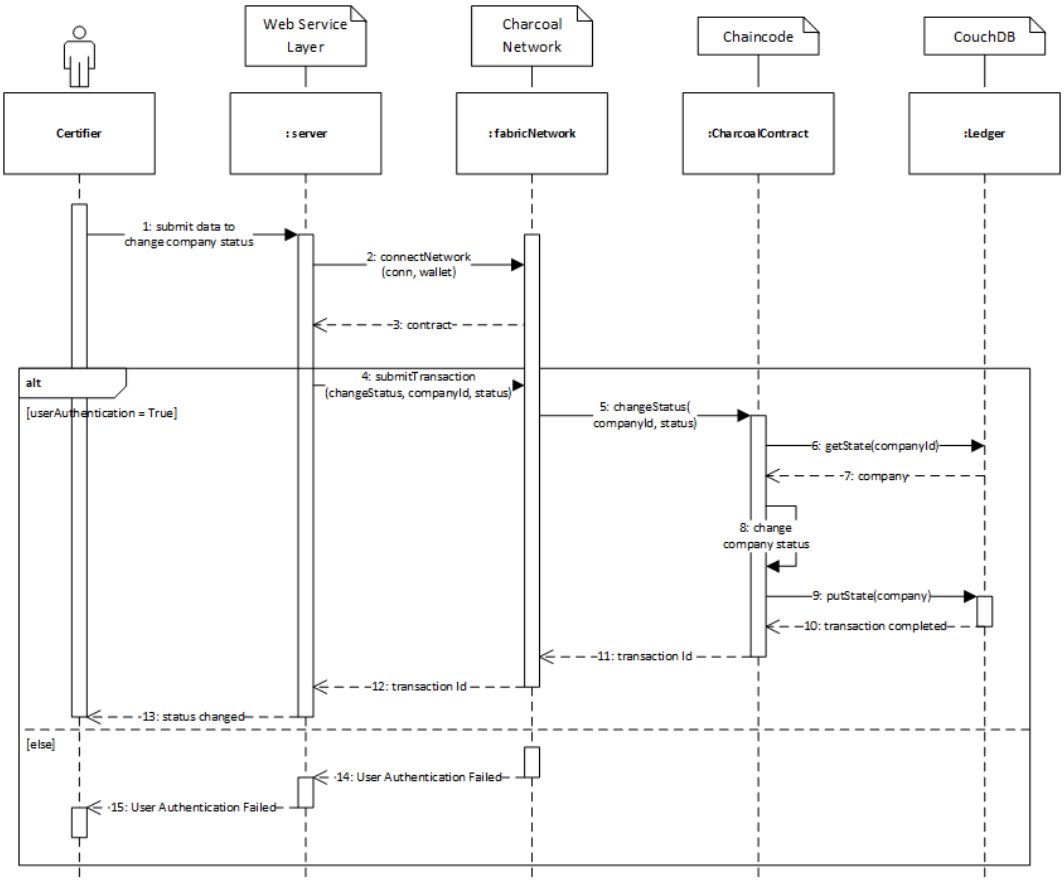

Figure 21. A sequence to change the company status

4.3.5.5 Perform Audit Sequence

The *Perform Audit* is one of the crucial processes, which is complexly modeled in a sequence diagram. It needs to be discussed in detail in order to understand the mechanism of audit process.

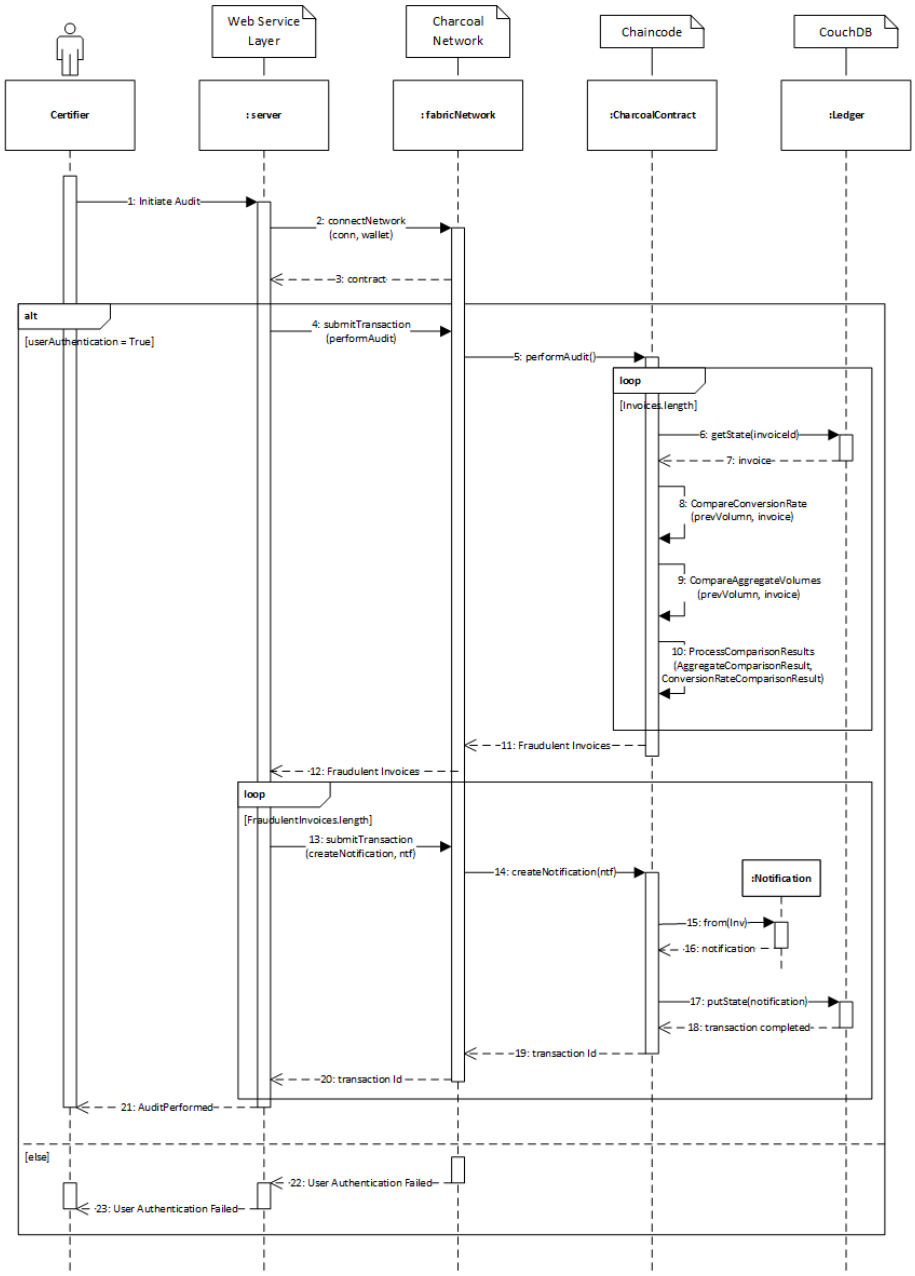

Figure 22. A sequence to initiate the audit process

The *Certifier* initiates the audit request. The chaincode fetches all the uploaded invoices from the ledger and iterates through all the invoices.There are three parts to this process, i.e., *CompareAggregateVolumes*, *CompareConversionRate*, and *ProcessComparisonResults*. to calculate these results, chaincode iterates through every updated state of a single invoice. See more detailed information about data and states of invoice in section 4.5.

First, discuss the *CompareAggregateVolumes*. This method takes two arguments, "prevVolumn" and "invoice" object, and returns a true or false value. The "prevVolumn" is considered as purchased volume by the seller. It calculates percentage aggregatedVolumn by dividing the conversion rate of seller into 100 and multiplying the result with "prevVolumn". If the "purchasedVolumn" is greater than the calculated "aggregatedVolumn", then the output will be true otherwise, it will return false. The true returning means there is a mis-compliance in the invoice, meaning that the seller is selling more volumn than he purchased.
Second method is *CompareConversionRate*, in which the implicit conversion rate be calculated by dividing sold "volumn" with "prevVolumn" and multiplying the result with 100. Then next step is to calculate the historic conversion rate of the seller by getting average results of all the conversion rates that the seller attempted in all the invoices. After getting the implicit conversion rate and historical conversion rate, it calculates the difference between them. Suppose the difference deviates from a certain threshold; this will be considered as mis-compliance, and the returning value will be true; otherwise, it will return false. If the seller's conversion rate is greater than 50, then the deviation should not be more than 5%, and if the conversion rate is below or equals 50, then the deviation should not be more than 2%. The sequence diagram of this process is shown in the figure 22.

4.3.6 Class Diagram

The ERD is the representation of tables stored in databases and the relationship between them. However, the class diagram represents classes and attributes of the system that will interact with each other. The class diagram derives from ERD and Sequence Diagram. Each action message in the sequence diagram converts into the class diagram methods, and the object converts into attributes.

There are six classes mentioned in the class diagram, i.e., *Invoice*, *Company*, *Certifier*, *Notification*, *BaseContract*, and *CharcoalContract*. The *Invoice*, *Company*, *Certifier*, and *Notification* classes are data models, representing the state that needs to store in the ledger. These classes have two methods; "from()" and "toBuffer()" which will convert the object from Buffer to object and vice-versa. The need for these methods is because blockchain stores the data in buffer, which is not a human-readable form. There is no concept of data types in the blockchain. Hence, the data from blockchain is converted into objects through these methods to utilize in the real world.

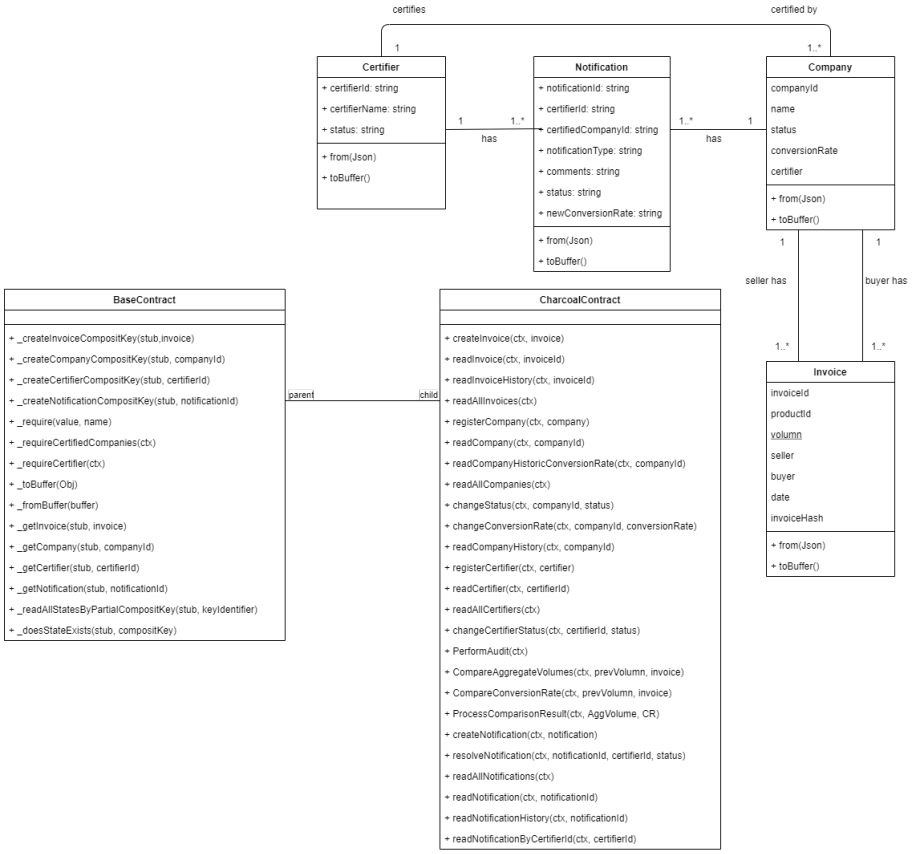

Figure 23. A class diagram illustrating a complete list of methods and attributes of smart contract

The *BaseContract* class is derived from the Contract class, which contains the most generic methods. The main class representing and have the chaincode logic is the *CharcoalContract* class. All the business logic and implementation of functionality are coded in this class. The client application calls the methods of this class to interact with blockchain.

4.4 Implementation

This section belongs to the fourth step of our research method that is Implementation and focuses on creating a hyperledger fabric from very scratch, starting from setting the operating system environment and installing prerequisites and tools. After setting up the environment, the next step will be the creation of a Hyperledger Fabric network. Everything will be discussed in detail further in separate sections.

4.4.1 Technology Stack

This section explains the technology stack I used, including operating systems, programming languages, and different tools and IDEs.

4.4.1.1 Operating System

The hyperledger fabric can be configured on Windows and Linux operating systems. However, it is easy and preferable to configure the Linux environment as compared to Windows. For the sake of implementation of hyperledger fabric, the Linux Operating system (Kali Linux) is selected. We can use any debian-based operating system, and the most popular and common are Ubuntu and Kali Linux.

4.4.1.2 Tools and Programming Languages

We need to set up a virtual environment using VMWare for development purposes, then install and configure (Kali Linux) operating system. Create a user with superuser privileges. There is a list of tools and dependencies required to be installed before configuring hyperledger fabric network.

1. Go (golang)
   Install Go (golang) first, the language in which hyperledger is written. Execute the following command in terminal to install.

   ```
   sudo apt-get install golang-go
   ```

   Configure the Go in path variable to globally recognise.

   ```
   export GOPATH=$HOME/go
   export PATH=$PATH:$GOPATH/bin
   ```

2. CURL
   Curl will help to call webservices and to download resources. Execute the following command in terminal to install.

   ```
   sudo apt-get install curl
   ```

3. NodeJs
   The hyperledger fabric smart contracts can be written in Go (golang), Python, Java and NodeJs. NodeJs is selected for writing the Chaincode because its is most familiar language among developers and easily understandable. Execute the following command in terminal to install.

   ```
   sudo apt-get install nodejs
   ```

4. Node Package Manager (NPM)
   NPM is required to execute the code written in Nodejs. Execute the following command in terminal to install.

```
sudo apt-get install npm
```

5. Python
   Execute the following command in terminal to install.

```
sudo apt-get install python
```

6. Docker
   Hyperledger Fabric is based on docker containers, we have to install these dependencies in order to run the fabric network. Execute the following command in terminal to install.

```
sudo apt-get install -y docker-ce
sudo apt-get install docker-compose
sudo apt-get upgrade
```

7. Hyperledger Fabric Binary Files
   The hyperledger fabric provides some binary files which are compulsory to create different components, e.g. create channel, registering organizations and many more. Execute the following command in Linux terminal to download the specified version of fabric binary files.

```
sudo curl -sSL https://goo.gl/6wtTN5 | sudo bash -s 1.4.3
```

   It will download the binary files from online resources into "fabric-samples" folder. All the further steps will be performed inside "fabric-samples" folder.

4.4.2 Blockchain Network Configuration

The hyperledger fabric is a decentralized distributed system consisting of several components that connect through an established network. While developing a hyplerledger fabric solution, there are three significant steps we should follow. First, establish a network architecture consisting of peer nodes, develop and deploy chain code, and develop a client-side web API to communicate with the network and perform transactions. A comprehensive network architecture diagram of the timber-to-charcoal audit process is present in figure 24 that describes different ledger components in a broader view and how they interlinked with each other.

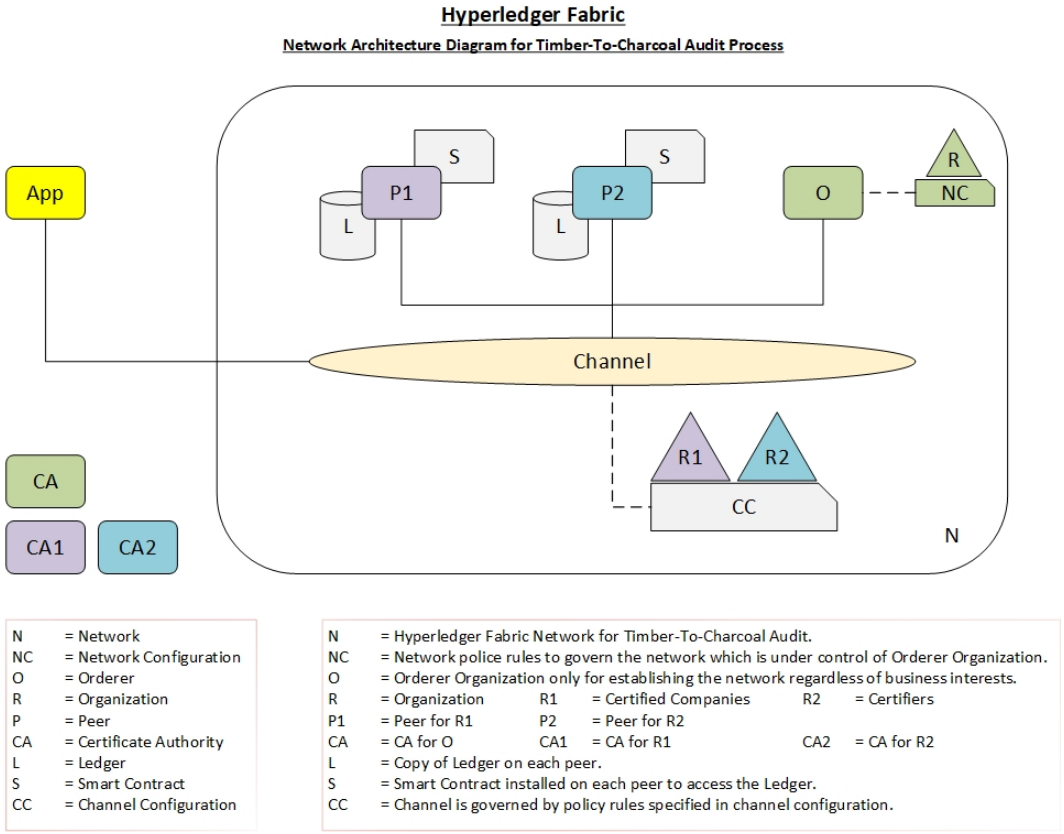

Figure 24. A network architecture diagram of hyperledger fabric for timber-to-charcoal process

The network configuration for the timber-to-charcoal audit process is defined in Figure 24. There is literal N representing a Network boundary consisting of different peers, a channel, and related configurations. The literal R represents an Orderer Organization responsible for maintaining the network policy rules written in NC network configuration deployed on Orderer peer node O. Orderer Organisation does not hold any ledger but maintains and assures each transaction transfer to other nodes within the network. The literal O is an orderer peer which holds the complete network configuration NC. There are two more organizations R1 representing Certified Companies and R2 representing Certifiers. Both the organizations have joined the same Channel because they need to perform their business transactions. The endorser peer node P1 is a physical node connected with the network for organization R1, the same as P2 for R2. Each endorser peer node has a local copy of the ledger stored and also has the same smart contract S agreed by both Organizations. Whenever a new organization joins the network, it will maintain a copy of the ledger and follow the same smart contract installed on other nodes within a channel. Each organization has a Certificate Authority (CA) which will provide

access control to authorized users of the network. After establishing a network, the clients need to communicate with the Fabric with the help of a client application. This client application is a web API that connects to the network and performs user requests. The process of creating the network is explained in detail further in this section.

4.4.2.1 Defining the Network

In order to start building the network according to the defined network architecture, we require some configuration files, ***crypto-config.yaml***, ***configtx.yaml***, ***docker-compose-couch.yaml***, and ***docker-compose-cli.yaml***. These files can be found in the charcoal-network folder when cloning the repository from GitHub inside ***fabric-samples*** folder with the help of following command.

```
git clone https://github.com/cybercommando/TimberToCharcoal-
    HyperledgerFabric.git
```

crypto-config.yaml: This file contains the definition of organizations. There is one orderer organization and two peer organizations, as mentioned in the network diagram.

configtx.yaml: This file contains the definition of organizations along with their MSP configurations. In this file, the organizations' name is entered into the consortium and channel to let the organization be a part of it.

docker-compose-cli.yaml: This file is responsible for starting the whole network. Everything will run inside the docker containers. It includes CA for every organization to manage the users who will communicate with them. It has a CLI container that provides the functionality to create channels and deploy chain codes to the peers joined within channels.

docker-compose-couch.yaml: This file is responsible for setting up the CouchDB as a database to store the ledger. By default, Hyperledger Fabric uses the LevelDB database, but the CouchDB supports more complex queries than LevelDB.

docker-compose-base.yaml: This file holds the properties of containers, including name, the port on which they will run, and volumes they utilize. This file automatically produces with the help of docker-compose-base-template.yaml file, when all the certificates are generated. This template file has variables that will replace with actual corresponding values to produce the original base file.

4.4.2.2 Generate Cryptographic Material

After the definition of network, digital certificates need to be generated to identify each organization as part of the network. The ***generate.sh*** file contains the script which will

generate the certificates for each organization with the help of binary files provided by Hyperledger Fabric. These binary files are located in the ***Bin*** folder within the ***fabric-samples*** folder.

```
../bin/cryptogen generate --config=./charcoal-network/crypto-config.
    yaml --output="./charcoal-network/crypto-config"
```

After executing this command, a folder ***crypto-config*** will automatically generate, containing all the organizations' certificates. It accepts the ***crypto-config.yaml*** file which contains the definition of organizations.

Another thing is to create the ***genesis block***. The purpose of the genesis block is to start the blockchain and connect it with the channel. The following scripts is mentioned in ***generate.sh*** script file that will generate the genesis block and channel description.

```
../bin/configtxgen -configPath ./charcoal-network -profile
    AuditOrdererGenesis -outputBlock ./charcoal-network/channel-
    artifacts/genesis.block

../bin/configtxgen -configPath ./charcoal-network -profile
    AuditChannel -outputCreateChannelTx ./charcoal-network/channel-
    artifacts/channel.tx -channelID mychannel
```

The script will use the ***configtxgen*** binary to generate the genesis block and channel inside the ***channel-artifacts*** folder.

4.4.2.3 Join Organizations to the Channel

After the definition of network and organizations corresponding with their certificates, the following steps are to run the docker containers, create a channel, and join all the organizations to that channel. The ***start.sh*** script file is responsible for this section.

Run Docker Container

In order to run the containers, the ***docker-compose*** command needs to be executed, which will receive the ***docker-compose-cli.yaml*** file as an argument, containing all the details about the containers. An additional file which is ***docker-compose-couch.yaml*** has the configuration to set up the CouchDB as a database to store the ledger. By default, Hyperledger Fabric uses the LevelDB database, but the CouchDB supports more complex queries than LevelDB. These scripts are written in the ***start.sh*** script file that can be seen in scripts folder inside a ***charcoal-network*** folder.

```
docker-compose -f ./charcoal-network/docker-compose-cli.yaml -f ./
    charcoal-network/docker-compose-couch.yaml up -d
```

Create a Channel

When all the containers are running, the next step is to create the channel with the help of CLI container. The docker exec command is used to create the channel which receives create channel scripts written in ***createChannel.sh*** file, which can be found in the charcoal-network/scripts/channel folder. The ***createChannel.sh*** contains the script to create the channel naming it mychannel and joining the first organization, the certifiedCompanies. The following script is formulated containing some variables defined before the script and used in it to create the channel.

```
peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME -f
    ./channel-artifacts/channel.tx --tls $CORE_PEER_TLS_ENABLED --
    cafile $ORDERER_CA >&log.txt
```

Another script follows this script to join the certified company into this channel.

```
peer channel join -b mychannel.block
```

Join Organization to the Channel

Once the channel is created and the certifiedCompanies organization is joined into the network, the certifiers organization also needs to join the network. Here the script file ***join-peer.sh*** is created, which will receive the parameter and dynamically fit into the script to install multiple organizations without code repetition. Inside the start.sh script file, executing the following command using join-peer.sh passing with required parameters.

```
docker exec -it cli ./scripts/channel/join-peer.sh peer0 certifiers
    CertifiersMSP 8051 1.0
```

This command needs to execute once per organization joining. The parameters are the name of the organization, the name of the MSP of the organization, the port number on which peer will run, and the version.

4.4.3 Chaincode

A network has been established at this step, a channel has been created, and both organizations have joined into the channel. The need is to develop the chain code and instantiate it on the network for deployment purposes. A separate NodeJs project is required with configuration defined in ***package.json*** file, which can be found in the chaincode folder in the repository. The ***index.js*** file is pointing out the chain code, which is written in CharcoalContract module.

According to the class diagram, all the chain code files have been generated inside the lib folder, discussed in sectoin 4.3.5. The figure 25 is illustrating the file hierarchy of all the classes placed inside the lib folder. Models folder contains the data model classes that represent objects that need to be stored in the ledger. Services folder class contains the abstract class and some helper modules used by the main charcoal contract class. The CharcoalContract is the main class pointed by the index.js, which implements all the functionalities of chaincode.

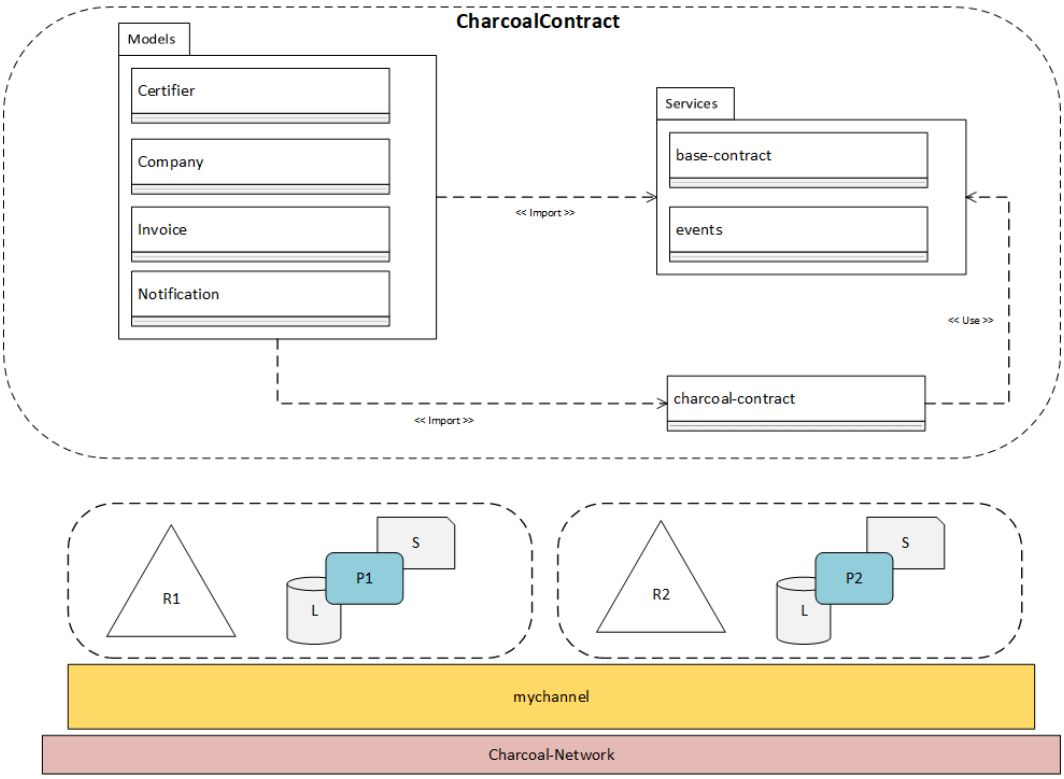

Figure 25. A chaincode class hierarchy deployed on the network

After developing chaincode, it needs to be installed on every peer node and finally instantiate on the network. The following command executes and installs the chain code on each peer.

```
docker exec -it cli ./scripts/install-cc/install-peer.sh peer0
    certifiedCompanies CertifiedCompaniesMSP 7051 1.0

docker exec -it cli ./scripts/install-cc/install-peer.sh peer0
    certifiers CertifiersMSP 8051 1.0
```

The command is referring to another file named ***install-peer.sh***. This file is dynamically accepting arguments and preparing the script required to install the chaincode. Once the chaincode install all the peers of each organization, the chaincode needs to instantiate on network with the help of following command written in ***install-cc.sh*** file. The ***instanciate.sh*** file contains the script to instantiate the chaincode on network.

```
docker exec -it cli ./scripts/install-cc/instanciate.sh
```

4.4.4 Web-services

A Hyperledger Fabric network has been established at this stage, joined by two organizations, chaincode has been deployed to each peer, and all the docker images running the entire network. The next step is to create a client application that will help users connect with the network and perform business transactions.

The NodeJs project is installed with Hyplerledger Fabric SDK libraries. Following is the list of libraries that are required to be installed inside this project.

1. fabric-network
   This library contains methods to submit transactions and make queries in the ledger. It contains Network related functions.

2. fabric-client
   This library contains methods to interact with channels, chaincodes, make queries, and monitor events in the ledger. It contains client related functions.

3. fabric-ca-client
   This library provides, the access control mechanism, the creation of users and their certificates, provoke and revoke access to a user.

4.4.4.1 Creating Users in the Network

One of the core features of Hyperledger Fabric is that it provides functionality to control user access. In order to communicate with the network, the end-user must be registered on the network. One administrator user is compulsory for each organization that will give permissions and access to other users.

The logic for enrolling the admin user is written in ***enrollAdmin.js*** file with specific steps. It establishes the connection to the organization and creates the CA client. Then it will create a wallet to store the certificates for the Administrator. Finally, it will create the admin's identity and save it into the wallet along with username and password.

When an admin user is registered on the network, the next step is to register an ordinary user to perform transactions. The ***registerUser.js*** file contains the logic with specific steps to create the users. It gets the connection to the organization, connecting the CA through gateway used by admins identity. Then create the user and saves its identity in the wallet.

4.4.4.2 Writing APIs

Now the connection to the network has been established, and some users exist who can communicate with the network. The next and final step is to create Restful web APIs, so the end-users consume them to interact with the hyperledger fabric.

There are two main files written to expose the API endpoints. The ***fabricNetwork.js*** contains the logic to connect with the network, and another one is ***server.js***, which contains the API methods. In ***server.js*** there are three types of request methods.

1. GET
   According to the provided parameters, these method types are used to create transactions to fetch the states stored in the ledger.

2. POST
   These method types are used to save the new data into the ledger. The data provided by these methods is validated first before saving it to the ledger and return the last transaction Id with a success message.

3. PUT
   These method types used to modify the state data. The data provided by these methods is also validated first before saving it to the ledger and return the last transaction Id with a success message.

The complete list of API endpoints is listed below in table 3.

Method	URI	Description	Request Params	Request Body	Response Body
POST	/api/initData/	To initialize sample data in ledger	none	none	JSON Response
GET	/api/performAudit/	To Initiate Audit Process	none	none	Invoice List
POST	/api/registerCompany/	To register a new company	none	Company	JSON Response
GET	/api/getAllCompanies/	This request will get a list of all the Companies	none	none	Company List
GET	/api/getCompany/	This request will get the company with ID	CompanyId	none	Company
GET	/api/readCompanyHistoricConversionRate/	This request will get the Historical Conversion Rate of the Company having ID	Id	none	JSON Response
PUT	/api/changeCompanyStatus	This request will change the company status	none	CompanyId, Status	JSON Response
GET	/api/getCompanyHistory/	This request will get the list of versions, of all the changes made in the Company data having ID	CompanyId	none	Company List
POST	/api/registerCertifier	This request will register a new certifier	none	Certifier	JSON Response
GET	/api/getAllCertifiers	This request will get a List of all Certifiers	none	none	Certifier List
GET	/api/getCertifier/	This request will get the certifier having Id	CertifierId	none	Certifier
PUT	/api/changeCertifierStatus	This request will change the Certifier status	none	CertifierID, Status	JSON Response
POST	/api/addInvoice	This request will upload an invoice data to the ledger	none	Invoice	JSON Response
GET	/api/getInvoice/	This request will get the Invoice having Id	InvoiceId	none	Invoice
GET	/api/getInvoiceHistory/	This request will get the list of versions of all the changes made in the Invoice data having ID	InvoiceId	none	Invoice List
GET	/api/getAllInvoices	This request will get the list of all Invoices	none	none	Invoice List
POST	/api/createNotification	This request will create a notification to change the conversion rate	none	Notification	JSON Response
PUT	/api/resolveNotification	This request will resolve a notification having notification Id	none	NotificationId, certifierId, status	JSON Response
GET	/api/getAllNotifications	This request will get all the notifications.	none	none	Notification List
GET	/api/getNotification/	This request will get a single notification having Id	NotificationId	none	Notification
GET	/api/getNotificationHistory	This request will get the list of version of changes made in Notification having ID	Id	none	Notification List
GET	/api/getNotificationByCertifierId/	This request will get the list of notifications associated with Certifier having Id	CertifierId	none	NotificationList

Table 3. List of API endpoints

4.4.5 Network Controller

The number of scripts are written in previous sections, performing different tasks, but they need to be simplified and organized to run in sequence to start the hyperledger fabric network. In order to do so, a ***network.sh*** script file is written, which will be responsible for starting the network, installing the dependencies, and stopping the network. A complete script call hierarchy is illustrated in Figure 26.

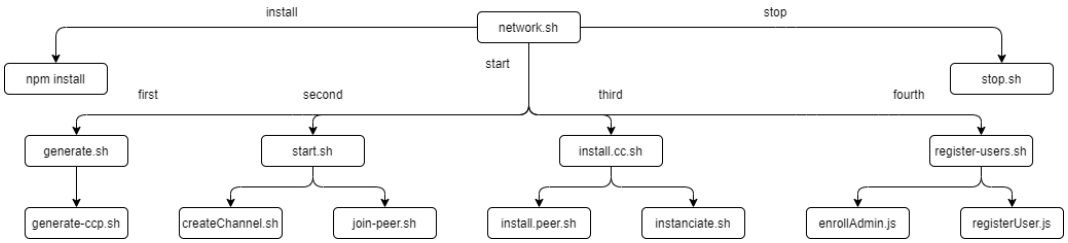

Figure 26. Script files call hierarchy

The following command will install all the node modules required for project:

```
sudo ./network.sh install
```

The following command will start the hyperledger fabric:

```
sudo ./network.sh start
```

The following command will stop the hyperledger fabric:

```
sudo ./network.sh stop
```

4.5 Sample Data Compilation

The hyperledger fabric saves the data in the form of key-value pairs in the world state ledger. A single state comprises two parts; the *Key*, which can be any identifier, and the *Value*, which holds any data structure [1]. Each state also maintains its version history. When data is inserted for the first time, it creates a state with revision 0. Each record inserted into the ledger for the same key will update the ledger state and increment the revision, as shown in Figure 27.

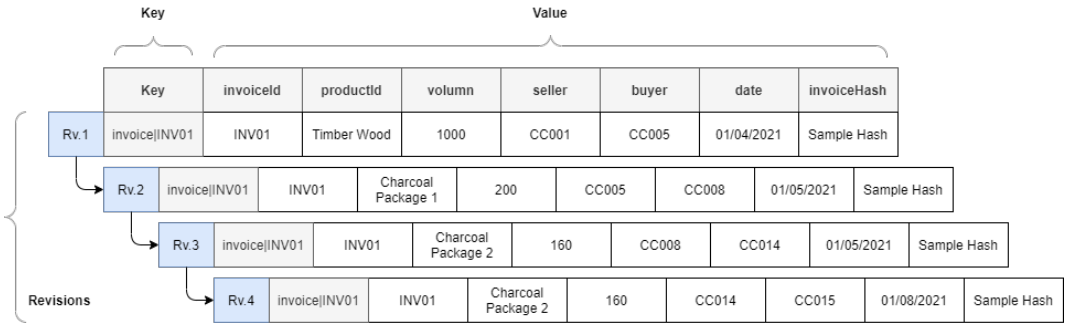

Figure 27. World state for single invoice data stored in the ledger

The following data set has been generated, considering world state structure.

4.5.1 Certifiers Data

Two certifiers enlisted in table 4 will certify the companies. The prefix for each Key stored in the ledger for certifier data is "certifier", e.g., "certifier_C01".

certifierId	**certifierName**	**status**
C01	Certifiers 01	ACTIVE
C02	Certifiers 02	INACTIVE

Table 4. Certifier Data

4.5.2 Certified Companies Data

A list of companies will perform transactions on the ledger, including uploading invoices and request to change their conversion rates. Some companies have an ACTIVE status, and some are already SUSPENDED, covering a test case that can not make transactions to the ledger. The prefix for each Key stored in the ledger for certified company data is "company", e.g., "company_CC001".The list of certified companies has mentioned in table 5.

companyId	name	status	conversionRate	certifier
CC001	Company 001 - Producer	ACTIVE	C01	100
CC002	Company 002 - Producer	ACTIVE	C01	100
CC003	Company 003 - Producer	ACTIVE	C01	100
CC004	Company 004 - Producer	SUSPENDED	C01	100
CC005	Company 005 - Charcoal Processor	ACTIVE	C01	20
CC006	Company 006 - Charcoal Processor	ACTIVE	C01	20
CC007	Company 007 - Charcoal Processor	SUSPENDED	C01	15
CC008	Company 008 - Secondary Charcoal Processor	ACTIVE	C02	80
CC009	Company 009 - Secondary Charcoal Processor	ACTIVE	C02	80
CC010	Company 010 - Secondary Charcoal Processor	ACTIVE	C02	80
CC011	Company 011 - Secondary Charcoal Processor	SUSPENDED	C02	65
CC012	Company 012 - Broker	ACTIVE	C02	100
CC013	Company 013 - Broker	ACTIVE	C02	100
CC014	Company 014 - Broker	ACTIVE	C02	100
CC015	Company 015 - Retailer	ACTIVE	C02	100

Table 5. Certified Company Data

4.5.3 Invoice Data

Invoice data is compiled so that once the first invoice data will upload, it will create a Key. For every subsequent invoice data for that specific invoice key, create a new version of data and automatically maintain the history of the invoice as explain in Figure 27. Some particular intentionally added mis-compliant records to cover the test case to identify the fraudulent behavior of sellers are shown in red colored marking in table 6.

INV09 should not be uploaded to ledger because both seller and buyer Certificate is Suspended. INV10 should not be uploaded because of Buyer Certificate status is Suspended. INV07 (Transaction 5) is illustrating the fraudulent behavior because the conversion rate should be 80%, but it is selling 110% of the volume. Hence It needs to be notified, and the status needs to be SUSPENDED after the Audit Process. INV08 (Transaction 5) is illustrating the fraudulent behavior because the conversion rate should be 80%, but it is selling only 65% of the volume. Hence It needs to be notified, and the status should be SUSPENDED after the Audit Process.

The prefix for each Key stored in the ledger for invoice data is "invoice", e.g., "invoice_INV01".

invoiceId	productId	volumn	seller	buyer	date	invoiceHash
INV01	Timber Wood	1000	CC001	CC005	01/04/2021	Sample Hash
INV01	Charcoal Package 1	190	CC005	CC008	01/05/2021	Sample Hash
INV01	Charcoal Package 2	152	CC008	CC014	01/05/2021	Sample Hash
INV01	Charcoal Package 2	152	CC014	CC015	01/08/2021	Sample Hash
INV02	Timber Wood	1000	CC001	CC005	01/12/2021	Sample Hash
INV02	Charcoal Package 1	200	CC005	CC008	01/13/2021	Sample Hash
INV02	Charcoal Package 2	158	CC008	CC015	01/15/2021	Sample Hash
INV03	Timber Wood	1000	CC001	CC006	01/18/2021	Sample Hash
INV03	Charcoal Package 1	200	CC006	CC013	01/19/2021	Sample Hash
INV03	Charcoal Package 1	200	CC013	CC009	01/19/2021	Sample Hash
INV03	Charcoal Package 2	154	CC009	CC015	01/20/2021	Sample Hash
INV04	Timber Wood	1000	CC002	CC006	01/25/2021	Sample Hash
INV04	Charcoal Package 1	200	CC006	CC008	01/26/2021	Sample Hash
INV04	Charcoal Package 2	152	CC008	CC015	01/27/2021	Sample Hash
INV05	Timber Wood	500	CC002	CC012	02/01/2021	Sample Hash
INV05	Timber Wood	500	CC012	CC006	02/02/2021	Sample Hash
INV05	Charcoal Package 1	95	CC006	CC009	02/11/2021	Sample Hash
INV05	Charcoal Package 2	76	CC009	CC015	02/12/2021	Sample Hash
INV06	Timber Wood	500	CC002	CC005	02/18/2021	Sample Hash
INV06	Charcoal Package 1	100	CC005	CC008	02/22/2021	Sample Hash
INV06	Charcoal Package 2	80	CC008	CC015	03/02/2021	Sample Hash
INV07	Timber Wood	500	CC003	CC006	03/02/2021	Sample Hash
INV07	Charcoal Package 1	100	CC006	CC009	03/10/2021	Sample Hash
INV07	Charcoal Package 2	110	CC009	CC015	03/12/2021	Sample Hash
INV08	Timber Wood	500	CC003	CC005	03/18/2021	Sample Hash
INV08	Charcoal Package 1	100	CC005	CC010	03/18/2021	Sample Hash
INV08	Charcoal Package 2	65	CC010	CC015	03/19/2021	Sample Hash
INV09	Timber Wood	2000	CC004	CC007	03/23/2021	Sample Hash
INV10	Timber Wood	2000	CC003	CC007	03/31/2021	Sample Hash

Table 6. Invoice Data

4.6 Testing

The implementation of developed solution needs to be tested to validate the outcomes according to defined test cases. This section refers to the testing of software solution, data immutability of hyperledger fabric.

4.6.1 Testing Solution Outcome

A test plan has been prepared with acute edge cases that are illustrated in figure 28. The test plan contains the test case Id to identify the test, the reference to the use case to which the test case belongs, a brief description of the test scenario, the steps that need to be performed for testing, the details of test data which is required for a particular test case, the expected test result, the actual results came after perform the particular test and result of test that is pass or fail.

Test Case ID	Use Case Reference	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Result (Pass/Fail)
TC01	UC01	Upload invoice data with buyer status ACTIVE and seller status ACTIVE	1) Create a webapi request using test data /api/addInvoice 2) submit the request.	invoiceId: INV11 productId: 10002 volumn: 1000 buyer: CC005 seller: CC001 date: 12/12/2021 invoiceHash: some temporary hash	Request response with success status code and Transaction Id.	As Expected	Pass
TC02	UC01	Upload invoice data with buyer status SUSPENDED and seller status SUSPENDED.	1) Create a webapi request using test data /api/addInvoice 2) submit the request.	invoiceId: INV09 productId: Timber Wood volumn: 2000 buyer: CC007 seller: CC004 date: 03/23/2021 invoiceHash: some temporary hash	Request response with Error Message.	As Expected	Pass
TC03	UC01	Upload invoice data with buyer status SUSPENDED and seller status ACTIVE.	1) Create a webapi request using test data /api/addInvoice 2) submit the request.	invoiceId: INV10 productId: Timber Wood volumn: 2000 buyer: CC007 seller: CC003 date: 03/31/2021 invoiceHash: some temporary hash	Request response with Error Message.	As Expected	Pass
TC04	UC01	Upload invoice data with missing parameters.	1) Create a webapi request using test data /api/addInvoice 2) submit the request.	invoiceId: productId: Timber Wood volumn: 2000 buyer: seller: CC003 date: 03/31/2021 invoiceHash:	Request response with Error Message.	As Expected	Pass
TC05	UC05	Perform the Audit	1) Create a webapi request using test data /api/performAudit 2) submit the request.	Invoice Data that compiled in Table No 5.	1) Request response contains the list of mis-compliant Invoices. (INV07, INV08) 2) Generate notifications against the sellers of these invoices.	As Expected	Pass

Figure 28. A list of test cases for testing the solution outcome

Each test case has been tested on the final solution to get the expected results. All the results have come up with the passed result that confirms the validity of the solution's output.

4.6.2 Testing Data Immutability

In this section, I will set up the hyperledger fabric network to test and examine the behavior of hyperledger fabric in case of data modification under the implication of different types of endorsement policies.

4.6.2.1 Establishing Test Environment

I require a multi-organization network to test the data tampering. I build the network of three organizations (CertifiedCompanies, Certifiers, and TestOrg). Each organization consists of one peer (peer0.Certifiers, peer0.CertifiedCompanies, and peer0.TestOrg), and each peer is installed with CouchDB to store the world state. There is one OrdererOrg and a CLI component to access the peer nodes. The graphical illustration of the network can be seen in figure 29.

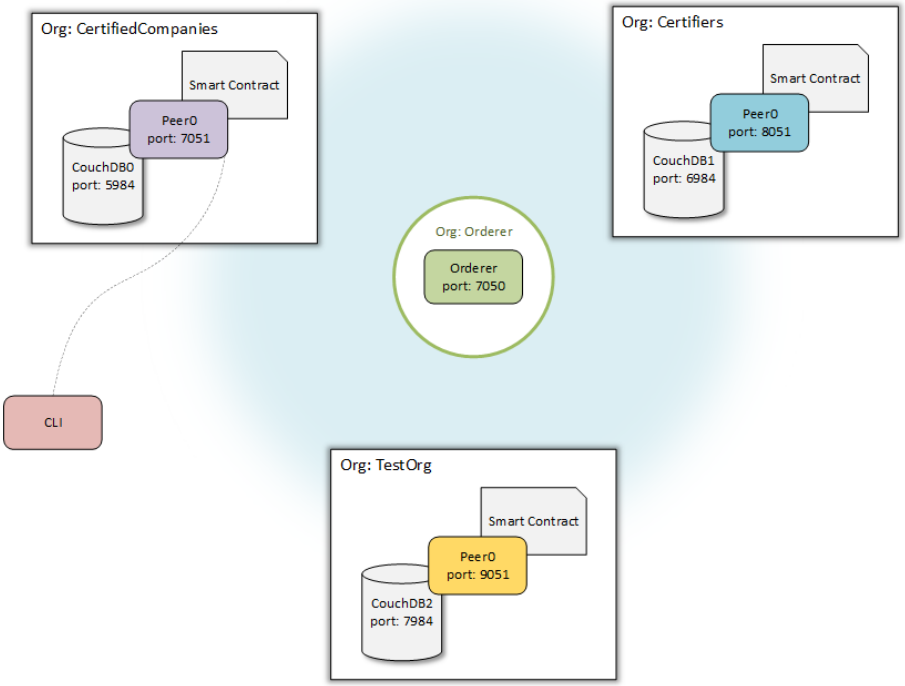

Figure 29. A network with three organization for testing the immutability of hyperledger fabric

This hyperledger fabric with the test network can be found in Tag v2.0 in the git repository mentioned in Appendix III. If we open the file ***instanciate.sh*** in the directory charcoal-network/scripts/install-cc we can see that by default, we set the endorsement policy to AND so that the transaction will validate when all the peers validate the transaction.

```
peer chaincode instantiate \
-o orderer.example.com:7050 \
--tls true \
--cafile $ORDERER_CA \
-C mychannel \
-l node \
```

```
-n charcoalcc \
-v 1.0 \
-c '{"Args":[]}' \
-P "AND ('CertifiersMSP.peer','CertifiedCompaniesMSP.peer','
    TestOrgMSP.peer')"
```

The network is configured and ready to start so that we proceed with testing scenarios. In order to start the network, run the following command and wait for running all the docker containers.

```
sudo ./network.sh start
```

The docker containers are ready, and all the peer nodes are running. Now we need to invoke the chaincode to initiate the ledger with default values. Run the following command to create a transaction in the ledger so that every peer will have an exact copy of the record in its CouchDB, as we can see in figure 30.

```
docker exec cli peer chaincode invoke \
-o orderer.example.com:7050 \
--tls \
--cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    ordererOrganizations/example.com/orderers/orderer.example.com/msp/
    tlscacerts/tlsca.example.com-cert.pem \
-C mychannel \
-n charcoalcc \
--peerAddresses peer0.certifiedCompanies.example.com:7051 \
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer
    /crypto/peerOrganizations/certifiedCompanies.example.com/peers/
    peer0.certifiedCompanies.example.com/tls/ca.crt \
--peerAddresses peer0.certifiers.example.com:8051 \
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer
    /crypto/peerOrganizations/certifiers.example.com/peers/peer0.
    certifiers.example.com/tls/ca.crt \
--peerAddresses peer0.testOrg.example.com:9051 \
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer
    /crypto/peerOrganizations/testOrg.example.com/peers/peer0.testOrg.
    example.com/tls/ca.crt \
-c '{"function":"initLedger", "Args":[]}'
```

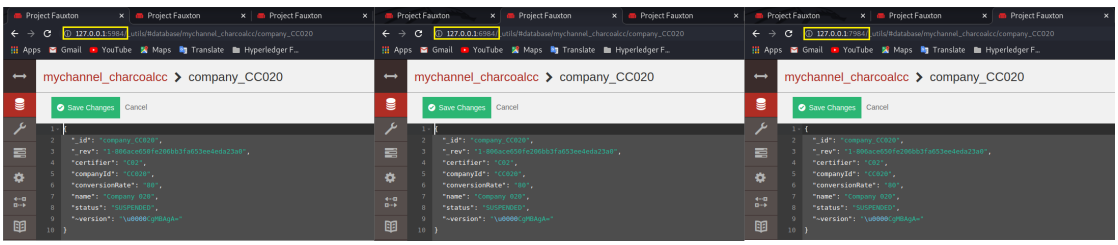

Figure 30. Data stored in each peer after initiating the ledger

4.6.2.2 Test Scenarios

I will test and observe three scenarios mentioned in table 7.

S/n	Test Scenario	Description
S1	Normal Transaction	In this case, the transaction will be performed without tampering the data in any node followed with AND endorsement policy.
S2	Transaction with data tampering (AND Policy)	In this case, the transaction will be performed when data in one of the nodes is tampered and chaincode is instantiated with AND endorsement policy.
S3	Transaction with data tampering (OR Policy)	In this case, the transaction will be performed when data in one of the nodes is tampered and chaincode is instantiated with OR endorsement policy.

Table 7. Different test scenarios to test data immutability

4.6.2.3 Test Scenario S1: Normal Transaction

All the peers and docker containers are up, and the chaincode is installed on each peer. We will first start testing the network with a standard transaction without considering any data tampering to check the system's behavior.

Step 1: Instantiating the chaincode on channel, with "AND(All Peers)" endorsement policy which means transaction will process when all the peers will validate the transaction and provide a valid response.

```
peer chaincode instantiate \
-o orderer.example.com:7050 \
--tls true \
--cafile $ORDERER_CA \
-C mychannel \
-l node \
-n charcoalcc \
-v 1.0 \
-c '{"Args":[]}' \
-P "AND ('CertifiersMSP.peer','CertifiedCompaniesMSP.peer','
    TestOrgMSP.peer')"
```

Step 2: Invoke the chaincode to initialize the ledger with default values. The default values can be seen in figure 30.

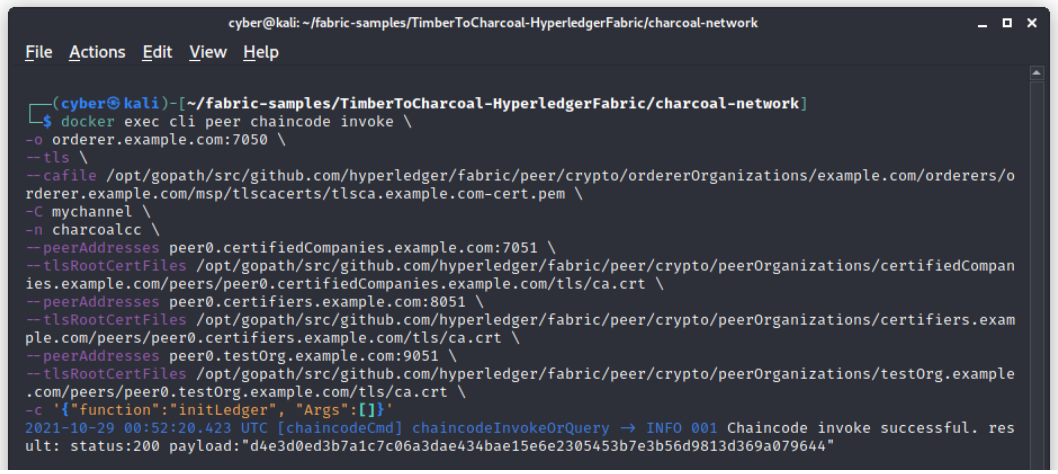

Figure 31. The chaincode invoked to initialise the default values

Step 3: We can query the ledger from all the peers to check that the data is identical on each peer.

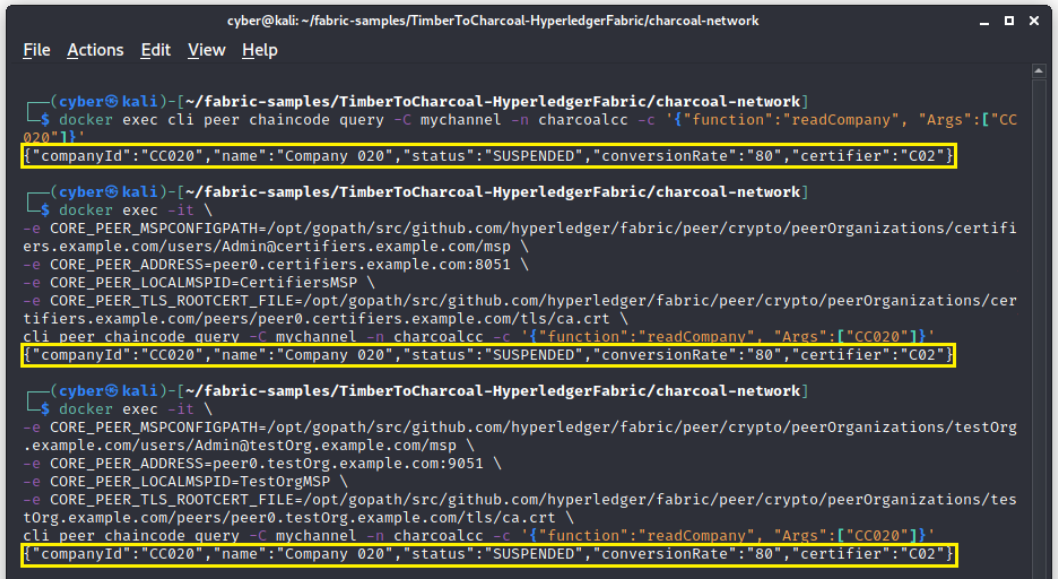

Figure 32. Query the ledger to see the identical data on each peer

Step 4: We will change the record by invoking the "changeConversionRate" chaincode and then again query the data to check. All the nodes have identical data and will validate the transaction according to the AND endorsement policy. We will query again to check the related change. As we can see in figure 33, the transaction has been endorsed, and the company conversion rate of the company has been changed from 80 to 82.

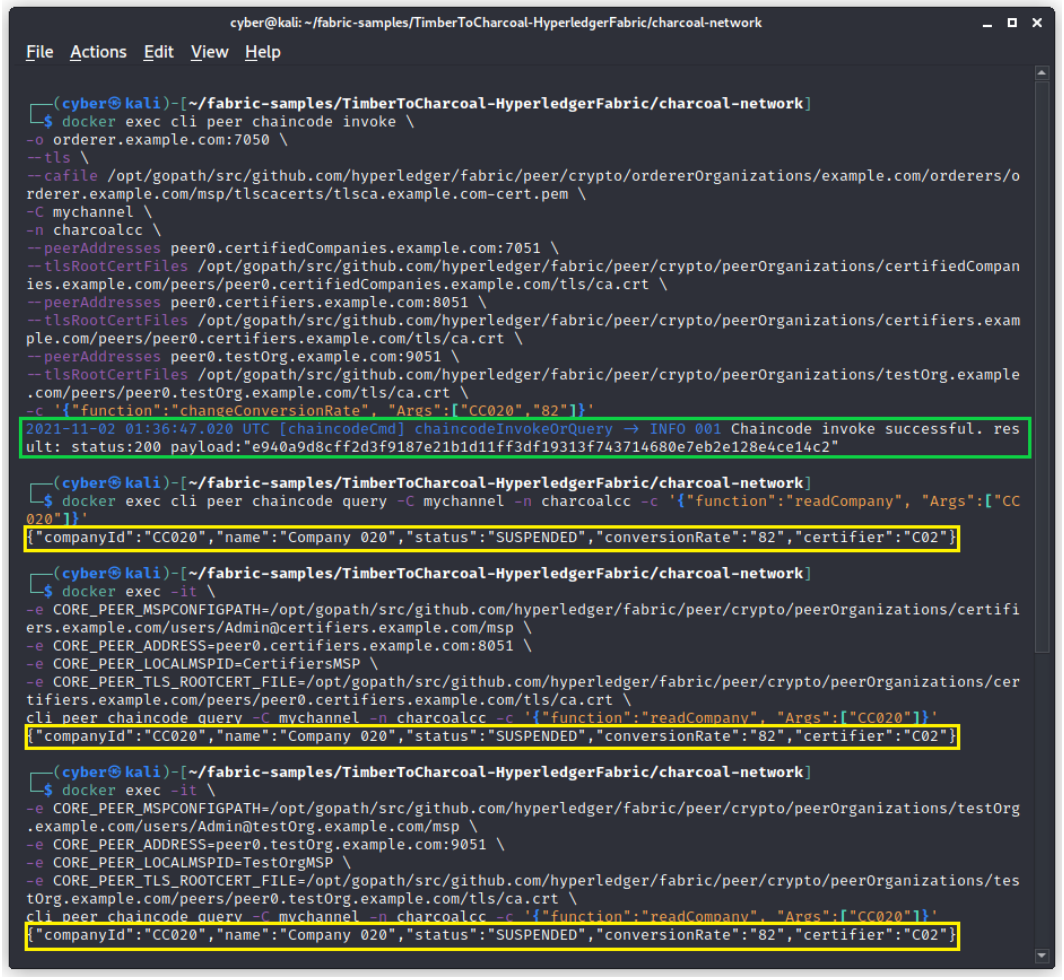

Figure 33. Invoke the chaincode to change the record and query data again to see changes

4.6.2.4 Test Scenario S2: Transaction with data tampering (AND Policy)

The steps in this test scenario will be the same as the previous test scenario. The only difference in this scenario is that we will access the world state of one of the nodes and change the data using a third-party tool.

According to the test environment, we have three CouchDbs(couchdb0, couchdb1, and couchdb2) for each peer running on ports 5984, 6984, and 7984, respectively. If we access one of the peers using the URL "http://127.0.0.1:5984/_utils/", we can see that database named "mychannel_charcoalcc" is present. We can access the record inside the database, change its content, and save it without knowing the hyperledger network.

The record with the id "company_CC020" is selected and changed its status value from

"SUSPENDED" to "ACTIVE" and saved in peer node "peer0.certifiedCcompanies" for intentional data tampering. We can see the change in figure 34.

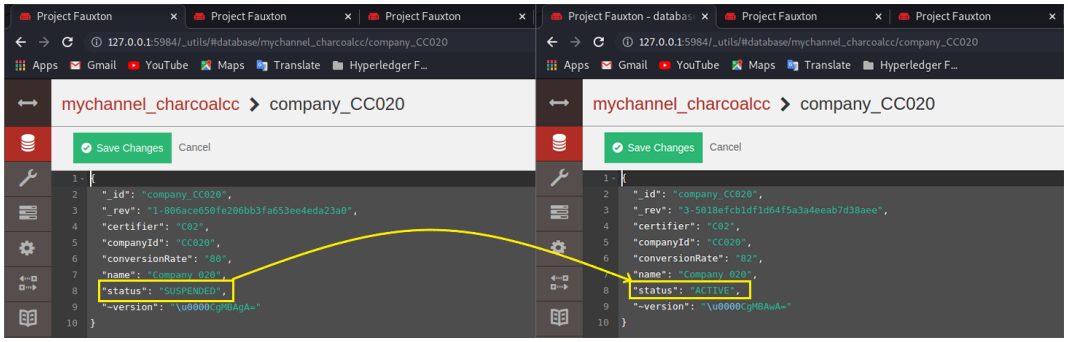

Figure 34. Changing the status of company

Step 1: Instantiating the chaincode on the channel.

Step 2: Invoke the chaincode to initialize the ledger with default values.

Step 3: Tampering data in world state of one of the peer nodes as shown in figure 34

Step 4: Querying the record from all the peers to see the different results.

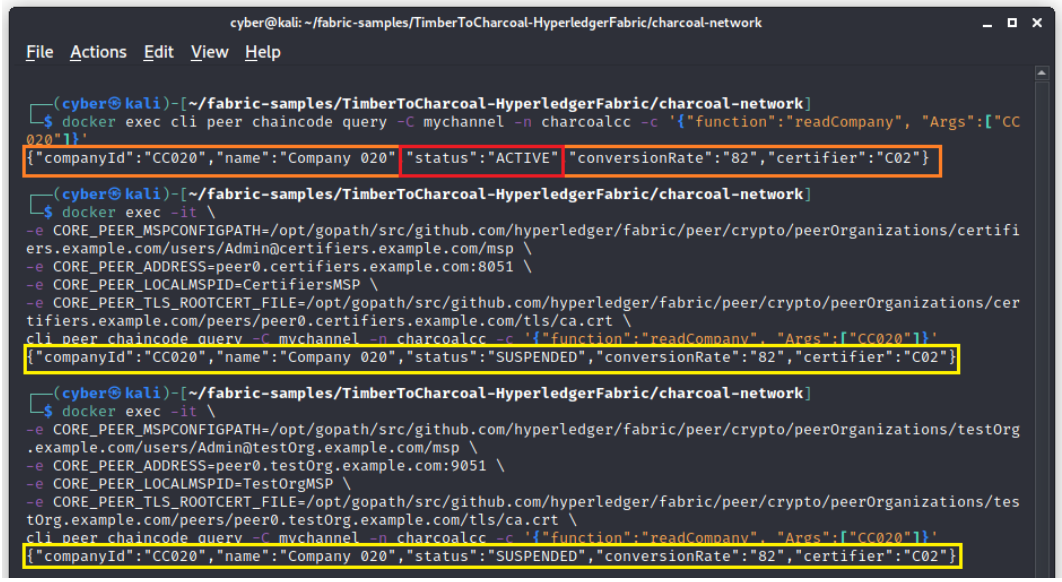

Figure 35. Querying the record from all peers to see the tampered node

Step 5: If we try to make a transaction to change the record, one tampered node is present in the network to decline the transaction. This will happen because of the endorsement policy we defined while instantiating the chaincode. The transaction will be validated if all the nodes endorse the transaction; otherwise, it will decline.

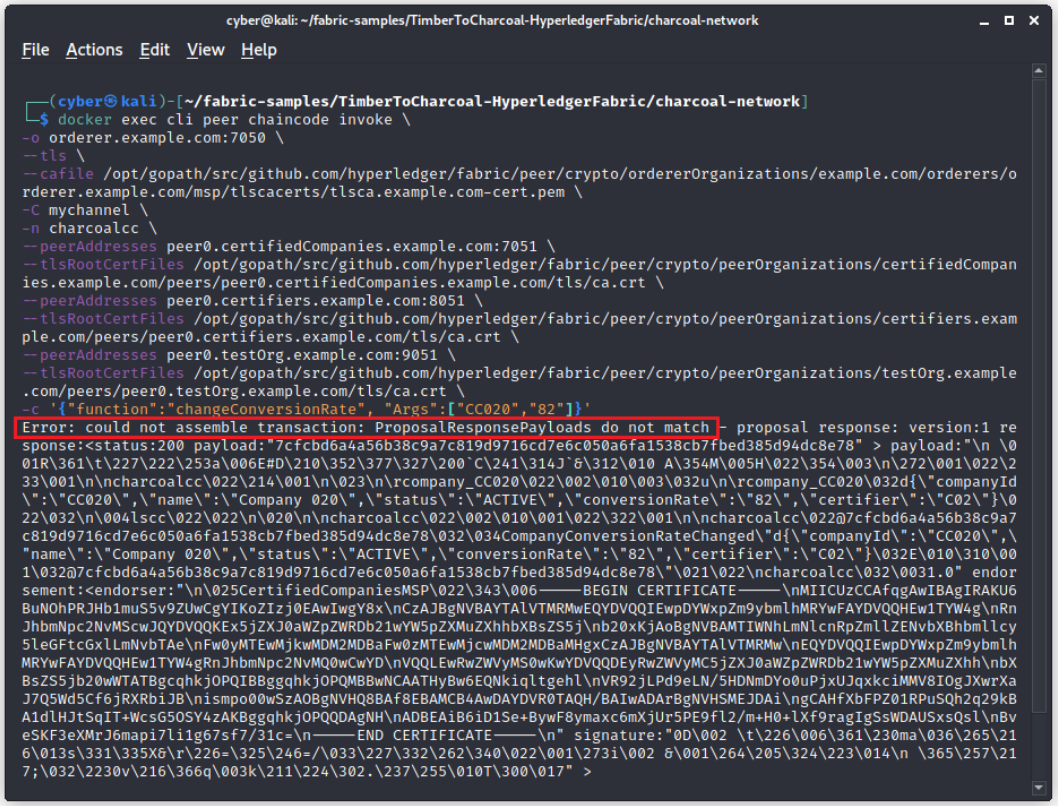

Figure 36. Transaction failed due to proposal response failure

We can see that only one of the nodes is compromised, resulting in transaction failure. I require this from the hyperledger fabric to help identify the data tampering and aid the audit process to proceed with honesty.

4.6.2.5 Test scenario S3: Transaction with data tampering (OR Policy)

We have tested the network by tampering with the record and followed by the AND endorsement policy. Let do a check if we have a weak endorsement policy. For this case, we will instantiate the chaincode with "OR(All Peers)" so that, if any of the peers endorse the transaction, the transaction will be successful, and tampered data will populate in the entire network. We will follow the same steps as before, except for the instantiation of

chaincode.

Step 1: Instantiating the chaincode on channel, with "OR(All Peers)" endorsement policy which means transaction will process when any of the peers will validate the transaction and provide a valid response.

```
peer chaincode instantiate \
-o orderer.example.com:7050 \
--tls true \
--cafile $ORDERER_CA \
-C mychannel \
-l node \
-n charcoalcc \
-v 1.0 \
-c '{"Args":[]}' \
-P "OR ('CertifiersMSP.peer','CertifiedCompaniesMSP.peer','TestOrgMSP
    .peer')"
```

Step 2: Invoke the chaincode to initialize the ledger with default values.

Step 3: Tampering data in world state of one of the peer nodes as shown in Figure 34

Step 4: Querying the record from all the peers to see the different results as shown in Figure 35

Step 5: We have one tampered node in the network, and according to endorsement policy, if any of the nodes endorse the transaction, the transaction should be validated and processed. We invoked a transaction to change the company's conversion rate in tampered node, and all the other nodes in the network propagated with the tampered data, as shown in figure 37.

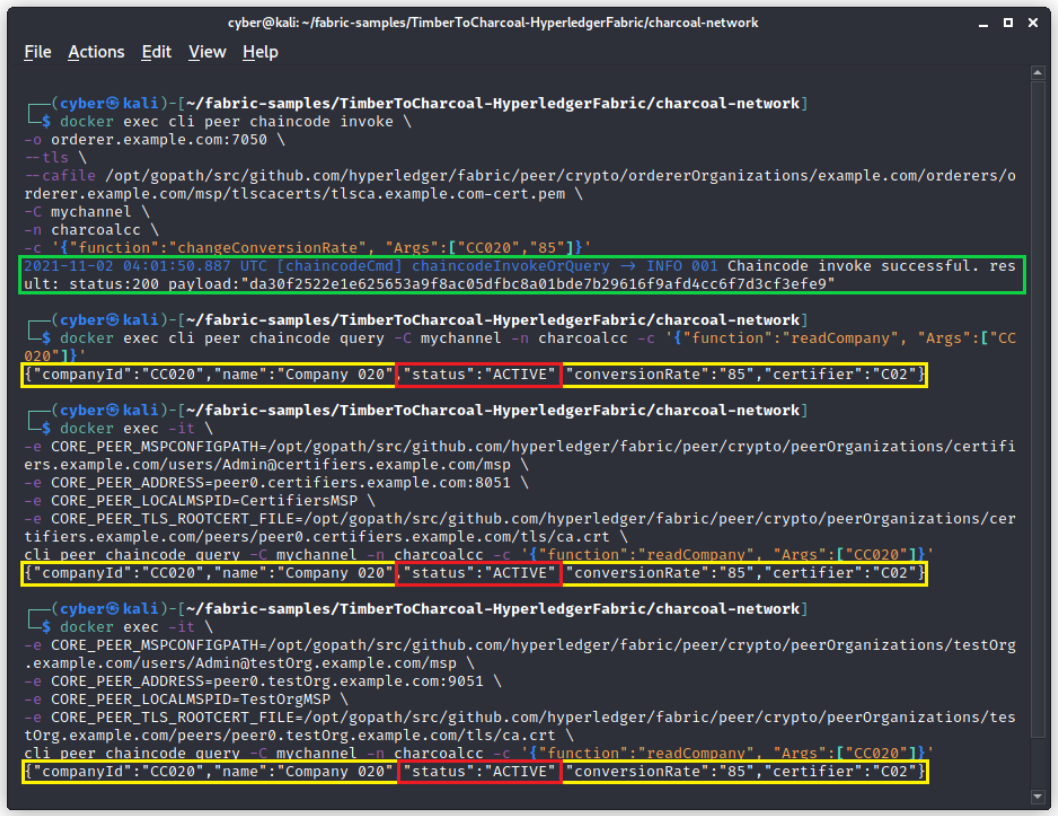

Figure 37. Tampered data propagated to entire network

4.6.2.6 Recovering Tampered Node

We notice that when there is a tampered node present in the network, the transaction cannot process until we fix the tampered node. To fix the node, we have to perform the following steps:

Step 1: Kill the docker containers running peer node of tampered and the related CouchDB. In our case we have peer node (peer0.certifiedCompanies.example.com) and CouchDB (CouchDB0). To kill the containers we have to run the following commands:

```
docker kill peer0.certifiedCompanies.example.com
docker kill couchDB
```

Step 2: Run the following command to bring these containers back online. When we run these commands, new containers are created, and CouchDB0 is initialized with a ledger copy identical to the untampered ledger in the network.

```
docker-compose \
-f ./charcoal-network/docker-compose-cli.yaml \
-f ./charcoal-network/docker-compose-couch.yaml up \
-d peer0.certifiedCompanies.example.com couchdb0
```

The docker containers start running again, and the blocked transaction can now process without any error.

4.7 Demonstration

The complete demonstration of the developed solution has been recorded in videos, and the reference links to videos can be seen in section 6.2 of this thesis. The Github repository also contains these videos. The documentation folder inside this repository contains JSDoc compiled documentation of the entire chaincode that can be view by running the index.html file in the browser.

The hyperledger fabric can only be called with the help of provided web API endpoint mentioned in section 4.4.4. When we start the network, we registered by default one admin and one user for each peer as shown in figure 38. Only these users can connect the network and invoke the chaincode.

```
*********************************
      Registering Users
*********************************
Wallet path: /home/cyber/fabric-samples/TimberToCharcoal-HyperledgerFabric/wallet/wallet-certifiedCompanies
Successfully enrolled admin user "admin" and imported it into the wallet
Wallet path: /home/cyber/fabric-samples/TimberToCharcoal-HyperledgerFabric/wallet/wallet-certifiers
Successfully enrolled admin user "admin" and imported it into the wallet
Wallet path: /home/cyber/fabric-samples/TimberToCharcoal-HyperledgerFabric/wallet/wallet-certifiedCompanies
Successfully registered and enrolled admin user "user1" and imported it into the wallet
Wallet path: /home/cyber/fabric-samples/TimberToCharcoal-HyperledgerFabric/wallet/wallet-certifiers
Successfully registered and enrolled admin user "user1" and imported it into the wallet
*********************************
       Starting API Server
*********************************

> timbertocharcoal@1.0.0 start
> node src/server/server.js

*********************************
API Server listening at localhost:3000
```

Figure 38. One admin and one user for each peer of each organization is registered to interact with the fabric network

4.8 Summary

In order to summarize this section, I conclude that this section covers the first four steps of our design science research method. The section 4.1 and 4.2 refer to the problem identification step, the first step in the design science method. In these sections, we discussed the case study of audit to the timber-to-charcoal business process. I came up

with the proposal of developing the blockchain-based solution to support cover the need and pain points of the process.

The section 4.3 refers to step two (Objectives of solution) and step three (Design and development) of the research method. In section 4.3.2 I formulated the core system requirements in user stories as the output step two. The rest of the subsections refers to the third step of the research method that is design and development. I developed software design artifacts including, Use cases, an Entity-relationship diagram, sequence diagrams, and a class diagram. However, I captured only the edge cases of the system and mentioned only essential features of the system in the design section.

The section 4.4 refers to the fourth step in the design science method that is Implementation. In this section, I first specified the tools and technologies required to develop the solution. Then I created the hyperledger fabric network, organizations and established an environment where further development was conducted. Once I defined the network, I developed the chaincode and deployed it in all the peer nodes connected. Then I compiled the sample test data in section 4.5 to test our defined solution in section 4.6. As the output of this fourth step, I finished the software development part and created some demonstration videos and documentation of the solution.

5 Evaluation

This section refers to the fifth step of our research method, which is the Evaluation. This chapter describes the evaluation criteria and discusses the results by mapping them on the outcome of the developed blockchain solution. This thesis implemented the case study of audit to the timber-to-charcoal business process. The primary purpose of the audit is to eliminate fraudulent behavior by preventing the users from manipulating the original data, getting the real-time data for accurate audit results, and faster detection of fraudulent behavior so that the auditor spends his valuable time in other aspects while auditing.

5.1 Evaluation Criteria

I defined the following evaluation criteria that cover and fulfill to evaluate the audit purposes that correlate with the features offered by hyperledger fabric.

5.1.1 Immutability

The capability of an organization to provide a complete and incontrovertible history of a transactional ledger simplifies and expedites the auditing process. The data must not be tampered, in order to yield reliable results while auditing the organization. Immutability term belongs to the feature of the system that helps to prevent the modification of data. Once the record enters the system, it is irreversible. For some reason, if a change is required, the previous record will always be there and never change. Instead of overwriting the record, a new entry will place into the system with proper versioning so that everyone can see the complete change history of a single record for clarity.

5.1.2 Decentralization

There are three primary types of network architectures exists that are centralized, decentralized, and distributed. Centralized network architecture contains one central server that performs all the processes and holds the data. Client application connects to the server and accesses and trusts the data whatever they receive. Distributed network architecture, on the other hand, functions similarly to centralized network architecture, except distributed servers are deployed in different locations. It aids the system in avoiding server overload. Decentralized network architectures have a different concept. Instead of having a single server responsible for making decisions and controlling the data, It enables the network to provide every individual with authority to govern the network with complete control without depending on a trusted third party.

5.1.3 Performance & Scalability

In the development of software systems, scalability is a crucial term. It refers to the increase or decrease of data and the system's ability to handle it without impacting system performance in terms of data manipulation and accessibility. The system should handle any change no matter what the circumstances and data processing demands are, without compromising the performance.

5.1.4 Security

When there is data and information, the next step is to secure that information from unauthorized access. Not only is the protection of data enough, but more factors need to be considered while securing data. One of the most popular approaches used in cyber security terms is the CIA triad [8]. This approach ensures the organizations about the security of valuable data without hindering other security parameters. This approach covers three essential principles (Confidentiality, Integrity, and Availability) that are discussed below.

5.1.4.1 Confidentiality

The confidentiality principle assures that only authorized bodies can communicate with the system and access the data. It is all about the access control mechanism for the system's users by implementing in several ways, e.g., username and password, some cryptography techniques or steganography, Etc.

5.1.4.2 Integrity

Integrity contributes to data reliability by guaranteeing that it is in the correct state and free of tampering. While ensuring that the data is protected against unauthorized access, still required by the system that the available data is correct and reliable or not. The integrity principle implements security protocols to ensure the authenticity of data, such as digital signatures or by hashing the data states. In the case of auditing the timber-to-charcoal, one of the significant challenges is getting tamper-proof data that can be obtained from the system by implementing the data integrity principle.

5.1.4.3 Availability

The availability principle ensures that authorized bodies have access to data at any time they need it. Compared to other principles, availability is more critical because if the data is not accessible by the authorized bodies, then the whole system is a waste. Availability is also hard to maintain because data is accessible through physical resources. Physical

resources can damage, need maintenance, require protection from natural disasters, and the most common problem the power failure.

5.1.5 Other challenges

With the change in business policies, needs, or customer requirements, software systems require improvements. Improvements can be a new feature, removal of an old feature, or fixing a bug. A system should be adaptable to receive improvements to fulfill the needs and survive within the innovative and growing software industry. As a good development environment, there should be enough technical support in documentation for the developers to feel native to the technology, maintain their interest in the field, and specialize quickly. In addition to facing development problems, we consider challenges like, technology improvement and lack of trust issues in specific technology by the organization

5.2 Evaluation of Results

In this section, I will evaluate the outcome of the solution on the basis of defined evaluation criteria mentioned in section 5.1.

5.2.1 Immutability

When we talk about blockchain, one of the most significant elements is data immutability. That is, once data is written in the blockchain, no one can change it. Specifically, the data are written in the blockchain data structure cannot be easily updated, and any attempt to do so can be promptly identified.

The consensus protocol of hyperledger fabric enables the entire network to store a complete copy of the ledger on each node connected to the network. The ledger consists of the world state and a blockchain. The world state consists of the current value, while the blockchain holds the transaction history. The transaction block can only be linked in the ledger (using a hashing mechanism) whenever the endorser node approves the node. However, the world state holds the current state of the data after performing all the transactions, which means, world state can be tempered. The consensus protocol of the Hyperledger Fabric ensures that the world state and the blockchain are the same.

As I discussed, each node of the network holds the complete copy of the ledger. The hyperledger fabric uses LevelDB by default as a database for the world state having lesser features than CouchDB. I selected CouchDB in the timber-to-charcoal blockchain audit network. CouchDB runs on separate docker containers and works independently as a third-party software so that it is easy to access the world state by connecting the

CouchDB and temper the record.

Suppose data has been tempered in the world state of any node, and the new transaction begins. In that case, the consensus mechanism of the Hyperledger Fabric will validate the transaction from other nodes. The response received from other data nodes will not match the current state of the tempered data node, resulting in transaction failure. The new transaction will never process unless we recover the compromised world state by replacing the entire ledger of that specific node with the network ledger.

5.2.2 Decentralization

The audit process of the timber-to-charcoal business involves multiple organizations. The certified companies upload their invoices in the blockchain, and certifiers perform the audit by accessing the invoice data uploaded by Certified Companies. Each node of certified companies and certifiers holds a complete ledger that gives data transparency to the whole network and meets the decentralized feature of Hyperledger Fabric.

Another feature of Hyperledger Fabric is creating channels to group up the organizations with the exact business needs. As I implemented in our solution, both parties (Certifiers and Certified Companies) must share data so that these organizations join the same channel.

As the business grows over time and new organizations join the network with different business needs. The making of private channels will aid them in partitioning the data and provide them with the separation of concerns. Every user in the network does not have access to every transaction. The fabric enables private transactions, which Ethereum and other blockchain-based technologies do not allow.

5.2.3 Performance & Scalability

Compared to other blockchain-based technologies, Hyperledger fabric does not implement the proof-of-work consensus algorithm or crypto mining, resulting in high scalability and fast transactions [23].

The transaction in Hyperledger Fabric has three abstract steps, including logic processing written in chain code, ordering the transaction throughout the network, finally validating and committing the transaction. It reduces overhead by ensuring fewer layers of trust and validation across different types of nodes.

A basic transaction processing flow in hyperledger fabric starts from submitting the transaction proposal to the endorser node. The endorsement policy defines the number of

endorsers required for the validation of the transaction. The endorser runs the chain code and responds with the signed endorsements back to the client. The client submits the transaction with all the signed endorsements, so the orderer node accepts the transaction circulates to the peer nodes. The peer node commits the transaction block to the ledger only when it confirms that the endorsement policy is met and there is no conflict in transactions. The limitation in the number of endorsers and the policy that only the clients have read/write signatures can traverse the network, optimizing scalability and increasing transaction performance in terms of speed.

However, Hyperledger Fabric is not the network fault-tolerant. Suppose the endorser fails to retrieve the endorsement from a defined number of peers due to network loss or some physical damage to the peer. In that case, the transaction will never be completed, resulting in transaction failure.

5.2.4 Security

The section holds the security concepts I implemented during the development of hyperledger fabric solution to the timber-to-charcoal audit process. I followed the CIA triad, the most widely used benchmarking evaluation criteria obeyed by the cyber security specialists to secure the information systems.

5.2.4.1 Confidentiality

Hyperledger Fabric is a permission blockchain. To control the unauthorized access, it came up with a feature known as Membership Service Provider (MSP). In the timber-to-charcoal blockchain, I have two organizations, certifiers and certified companies. While defining the network, I created Certificate Authority for each organization.The Certificate Authority (CA) is responsible for generating and issuing certificates to every user who needs to interact with the network. Users can be assigned roles such as "client" or "admin" to restrict access to the network. At least one admin user is compulsory in the network who will be further responsible for enrolling more users.

Hyperledger Fabric provides two types of policy types for the sake of access control, as described below:

1. SignaturePolicy
   Signature-based policies are more complex and return a boolean decision based on the combination of MSP principles. I defined the rules in ***configtx.yaml*** file for the timber-to-charcoal business process for each organization I registered.

Organizations:

```
- &OrdererOrg
    Name: OrdererOrg
    ID: OrdererMSP
    MSPDir: crypto-config/ordererOrganizations/example.
        com/msp
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Writers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Admins:
            Type: Signature
            Rule: "OR('OrdererMSP.admin')"

- &CertifiedCompanies
    Name: CertifiedCompaniesMSP
    ID: CertifiedCompaniesMSP
    MSPDir: crypto-config/peerOrganizations/
        certifiedCompanies.example.com/msp
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('CertifiedCompaniesMSP.admin', '
                CertifiedCompaniesMSP.peer', '
                CertifiedCompaniesMSP.client')"
        Writers:
            Type: Signature
            Rule: "OR('CertifiedCompaniesMSP.admin', '
                CertifiedCompaniesMSP.client')"
        Admins:
            Type: Signature
            Rule: "OR('CertifiedCompaniesMSP.admin')"

- &Certifiers
    Name: CertifiersMSP
    ID: CertifiersMSP
    MSPDir: crypto-config/peerOrganizations/certifiers.
        example.com/msp
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('CertifiersMSP.admin', '
                CertifiersMSP.peer', 'CertifiersMSP.
                client')"
        Writers:
            Type: Signature
            Rule: "OR('CertifiersMSP.admin', '
```

```
                CertifiersMSP.client')"
        Admins:
            Type: Signature
            Rule: "OR('CertifiersMSP.admin')"
```

These policies can be defined in multiple locations. The most common way is to define in the ***configtx.yaml*** file as I did, but sometimes the policies can be defined in the CLI command when instantiating the chain code using -P parameter as we can see in following command. These commands work as a boolean statement, which allows us to write complex boolean statements according to our business policies.

```
peer chaincode instantiate \
-o orderer.example.com:7050 \
--tls true \
--cafile $ORDERER_CA \
-C mychannel \
-l node \
-n charcoalcc \
-v 1.0 \
-c '{"Args":[]}' \
-P "AND ('CertifiersMSP.peer','CertifiedCompaniesMSP.peer','
    TestOrgMSP.peer')"
```

2. ImplicitMetaPolicy
   These policies are more generic and set as default for the entire network. These policies are only defined in the configuration in broader ways to govern the access in the application section of the configtx.yaml file.

```
    Application: &ApplicationDefaults
    Organizations:
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
```

The criticality of endorsement policy depends upon the nature of organizations and the business rules. We can define the policies as listed above to achieve a certain level of validation. If we require a higher level of validation, we can define that majority of the

organization members validate the node and get the transaction validated.

Hyperledger fabric supports Transport Layer Security (TLS) to secure the transactions between nodes. It helps to avoid the man-in-the-middle attack and secure the communication between nodes throughout the network. A single node behaves TLS server when it receives the transaction and behaves TLS client when it sends the transaction.

5.2.4.2 Integrity

The endorser nodes in hyperledger fabric provide data integrity by ensuring that data is not consumed twice. In hyperledger fabric, the orderer service is responsible for validating the transaction from each peer connected to the network. This policy is governed by the consensus mechanism used by hyperledger fabric [34]. It also ensures that data will remain the same in all nodes throughout the execution. In the case of data temper, the transaction is held, and commit is not allowed until the tempered resolves. The hyperledger fabric provides functionality to define the endorsement policy for every chaincode deployed. Endorsement policy defines the criteria for how many numbers of endorsement responses are required to validate the transaction.

5.2.4.3 Availability

The hyperledger fabric enables the trusted parties to perform the transaction through peer-to-peer distributed network [31]. And they are more faster than centralized systems [27], which ensures that the network is always online. There is no single point of failure; even if one node stop responding, the rest of the nodes will maintain the network, and the information can still be accessed without any data loss. All the nodes have a complete ledger, and nodes communicate with each other to keep data up-to-date and always available.

Cyber Attacks such as Distributed Denial of Service (DDOS) cause centralized applications to stop responding, resulting in loss of financial values [26]. The hyperledger fabric follows concensus mechanism in which 51% of the entire network required to take down in order to fail the network. The hyperledger fabric is able to define multiple and complex policies as I stated in section 5.2.4.1, in order to defend the attacks. We can increase the number of endorsement percentage to decrease the network exploitation rate.

5.2.5 Other Challenges

In this section, I will discuss the challenges faced by hyperledger fabric as an emerging technology.

5.2.5.1 Implementation and Support

I considered writing user stories before start designing the network. However, the development of the blockchain-based solution is not conventional, but the designing of software engineering artifacts can be similar to conventional software development. Different software engineering artifacts have been developed to understand the users, processes, workflow, entities of the system, and in-depth functionalities of the system. It shows that we can consider standard software design artifacts and instruct the developers to develop the system.

The smart contracts in the blockchain are written in some programming languages. Hyperledger Fabric supports the most common and basic programming languages (Go, Java, Javascript, Kotlin) so that a large number of programmers can start developing it without shifting their expertise or spending more time to learn before starting development. The chain code for the timber-to-charcoal audit network and the rest full web APIs are written using NodeJs.

In Hyperledger fabric, we only write the chain code in the programming language. The configuration of the network, defining channels and organizations, establishing policies, and deploying chain code in the network are all written in scripts that run on *Docker Compose CLI*. It is an only pain point for the Hyperledger Fabric developers. A high-level understanding of networks and DevOps is required in order to configure the network. Configuring the network is not as complex as maintaining the network. As the network grows and new rules come up to be implemented, the network needs some maintenance and requires modification. If we require to change the network configuration file, we need to rewrite the configurations and run the script to update with proper versioning. We also set up the modification policy specific to admins or members of the orderer organization or whatever the defined policy is. Only these nodes can implement the change. Once the configuration updates, it synchronizes with the entire network.

Deployment of the chain code follows the same procedure. New chain code is written and then instantiated to the network, maintaining the version history and synchronize with the network. Whenever there is a change in the network, we have to rewrite the configuration files and upload them via CLI commands.

While developing the timber-to-charcoal network, Lot of problems occurred in searching for technical assistance. Resolving a bug took much time by searching it over the internet due to a lack of community support. There are significantly fewer results found on the internet related to the problem. Hyperledger Fabric has complete documentation available on the internet, but it is not straightforward and requires much time to understand.

5.2.5.2 Quantum Computing

Blockchain is known for its decentralization and high level of the encryption mechanism. No doubt, it is significantly hard to break the encryption of Blockchain with currently available technologies. However, the new era of quantum computing is coming with faster computing and significantly improved algorithms that can perform reverse engineering on encryption techniques to break them [33]. Currently, quantum computing is not so common and not be possible to use, but it can impact blockchain cryptography which is the biggest challenge for the security concept of Blockchain.

5.2.5.3 Complex Architecture

Every new technology comes up with various benefits and also along with drawbacks as well. The hyperledger fabric is new and faces many challenges involving lack of developers, lack of community support, and the most important, complexity in understanding and implementation.

5.2.5.4 Trust Issues

Technology shift in any organization involves a risk of a trust issue with new technologies. The biggest concern of any organization is to safeguard its assets and confidential documents. They cannot afford any data loss or put their valuable time into new technology. It requires many implementations and real-world applications in a working state or at least tested and validated as I did in this study. These studies are mandatory to gain trust so that more businesses trust it and take a step forward to implement new technology.

5.2.5.5 Lack of Automated Time-based Events

In the timber-to-charcoal process, the audit process needs to be performed periodically. The audit process should initiate automatically after two weeks when enough invoices are uploaded into the ledger, as stated in the business process model mentioned in figure 8. However, to achieve this requirement, I need a time-based automated event that will trigger automatically after two weeks. Unfortunately, during the development of hyperledger fabric chaincode, I determined that hyperledger fabric does not provide any support or API that helps us achieve automation to invoke the chaincode after a certain period.

5.2.5.6 Energy Consumption

The blockchain provides availability to the business assets, as all the peers hold a complete copy of the ledger. Bitcoin and Ethereum use a proof-of-work consensus algorithm, in which all the peers try to solve the problem to validate the transaction. It requires a lot

of computational power and energy resource, which is quite costly to manage [9]. However, hyperledger fabric does not implement a proof-of-work consensus algorithm but still requires 51% of approvals from connected peers [27]. This proportion will never affect a smaller network, but considering more extensive networks is a point to worry about.

Hyperledger fabric is immutable so that once the record is created, it will never delete, resulting in an extensive amount of data storage required by every peer connected to the network. Those organizations who have to connect and perform a single transaction suffer to manage the entire ledger of the network. In this case, organizations can refuse to prefer this solution.

5.2.5.7 Laws Imposed by Government

The backbone of all cryptocurrencies is blockchain technology. The blockchain is not serving only cryptocurrencies but also other business solutions, as I discussed in this study. Hyperledger Fabric is one of the emerging technology bases of blockchain technology. However, many countries ban the blockchain technologies specifically cryptocurrencies e.g., Algeria, Egypt, Moroco, bolovia, Nepal, and China. Some countries only ban banking and financial services from operating [36]. In this situation, any change in governmental policies, laws, and regulations may affect the proposed solution's validity.

5.3 Summary

This chapter consists of three sections. In the first section, I defined the evaluation criteria on which the developed blockchain-based solution will evaluate. I searched and could not find any specific evaluation criteria to evaluate the blockchain-based enterprise software solution. Hence, I defined our own criteria, considering the need for the solution and pain points mentioned in sections 4.1 and 4.2. Then I evaluated the solution outcome with the defined criteria and discussed several pros of cons of using a blockchain-based solution.

In the third section, I discussed the limitations and threats to the validity of using blockchain-based solutions. This chapter refers to the fifth step of the design science research method and addresses our research question (RQ2). According to the design science methodology, we can iterate again and again to improve the solution based on findings and measures achieved by evaluation after evaluating the solution. However, I considered only one iteration and proceeded to conclude the study results.

6 Discussion

In this section, I discuss the results and outcomes of the study in the context of our research question. The table 8 explains the general mapping of our study to our defined research questions.

RQ no.	RQ Description	Section addressing RQ
RQ1	How to build blockchain-based solution to audit the timber-to-charcoal process?	Section 4 Case Study
RQ2	How the blockchain-based solution can help the auditing of timber-to-charcoal process?	Section 5 Evalution

Table 8. Mapping of research questions on study

The RQ1 focuses on the development of the timber-to-charcoal process. In order to develop the blockchain-based solution, in section 4, I first discussed the case study of the timber-to-charcoal process in detail. After analyzing the case study, I extracted requirements in the form of user stories as mentioned in section 4.3.2. The user stores helped identify the users of the system and the exact core requirement of the system. Once I have a clear view of core requirements as user stories, I developed the use case diagram to identify the system boundary and the user interaction to the system's feature, as discussed in section 9. The use case diagram illustrated the actors and use cases, which then further explained in detail individually, defining preconditions, postconditions, and sequence of interaction with exceptions.

I defined an ERD diagram to understand the data models to create the structure of the ledger. I came up with four data models that need to be stored in the ledger, i.e., Certifier, Company (certified companies), invoice (uploaded certified companies, on which certifiers perform audit), and the Notifications generated by certifiers to change the conversion rate and by the audit process to identify the certifier about the fraudulent behavior captured during the audit.

With the help of the sequence diagram mentioned in section 4.3.5, I linked the data models and the sequence flow mentioned in use cases and created sequence diagrams. The sequence diagram illustrates the internal system calls and allows the reader to see how the chaincode invokes. I have all deeper understanding of the system, and I finalized the class diagram to write the chaincode.

To implement the hyperledger fabric solution, I first defined the network, consisting of organizations and peers, defining the consensus protocols, and creating identities and

policies for those organizations to join the network. Then I wrote the chaincode and developed it on every peer which is connected with the network, as we explained in section 4.4.3.

When the network is running, and chaincode is installed on every peer node, I created web APIs to interact with the network to invoke chaincode mentioned in section 4.4.4. Any authenticated user can consume these web API endpoints. I stop at this stage, but the readers can further enhance the code by creating user interfaces in their favorite frontend language to interact with the ledger network.

The RQ2 focuses on the outcome of the developed solution and our findings in terms of pros and cons. The section 5 addresses this question and provides the benefits of using hyperledger fabric. First, I defined the evaluation criteria as described in section 5.1. Secondly, I mapped the outcome and discussed the results following the criteria in section 5.2. The table 9 is illustrating the pros and cons I determined while implementing the solution for auditing the timber-to-charcoal process.

Pros	**Mapping with study results**
Immutable ledger	It provides aid to the audit process and guarantees that data does not tamper with fake values.
Decentralization	Data is not present in one central server so that everyone has an exact copy of the data and they validate it.
Performance and scalability	The hyperledger fabric doesn't use the proof-of-work consensus algorithm resulting in a lower transaction time. It eliminates the time consumption by the auditors during the audit process.
Confidentiality	The involvement of different organizations requires confidentiality to their assets and records. The hyperledger fabric provides MSP, which prevents the network from unauthorized access. The channels in hyperledger fabric provide the facility to the network to group up the organizations with similar concerns.
Integritiy	The endorser node assures that the transaction is circulated in all the peers in the network so that they validate the transaction, and it is also responsible that data is not consumed twice.
Availability	Every peer node connected to the network contains a copy of the ledger. If any of the nodes fail or crash, data is still accessible throughout the network.
Cons	
Lack of support	There is very limited support available that helps in developing the hyperledger fabric solution.
Quantum computing	Quantum computing computes very complex algorithms, which is a threat to current complex encryption technology.
Complex Architecture	The development of hyperledger fabric requires a skilled professional who has a broader knowledge about computing and network infrastructure.
Trust Issues on new technology	Well-established organizations don't invest time and effort into emerging technologies due to immature trustworthiness.

Table 9. Pros and cons of using hyperledger fabric

6.1 Future Work

In this thesis, I developed the hyperledger fabric-based decentralised application for audit to timber-to-charcoal. The performance of hyperledger fabric-based application is evaluated by using the descriptive evaluation method suggested by the design science research method [25]. The future work is to perform performance benchmarking on the developed solution to calculate the transaction's speed, latency, and throughput. The aim is to test the developed network using an hyperledger benchmarking tool (Caliper). It possible to configure the network, develop benchmarking scripts and run the scripts to obtain results. The performance benchmarking can help the work improve the developed solution and aid the audit process for the timber-to-charcoal business process.

6.2 Resources

The code snippet used throughout the thesis can be found in my public git repository. The demonstration videos have been created that represent the working of our developed solution.

6.2.1 Github Repository

There are two code releases in this repository. The release v1.0 has a complete implementation of hyperledger fabric for audit to timber-to-charcoal network. It consists of two organization, one for CertifiedCompanies and the second one for Certifiers. It has complete chaincode implementation for timber-to-charcoal network. The release v2.0 contains the code with three organizations. This release specifically created to perform testing only. The data immutability test is perform in this code. The link to the repository is as follows:

`https://github.com/cybercommando/TimberToCharcoal-HyperledgerFabric.git`

6.2.2 Demonstration Videos

There are demonstration videos to help readers see how to operate the software solution and how it works. The purpose of creating videos is to reduce the time of readers. They can see the code implementation without configuring the system and running the code.

Start Network and Performing Audit
`https://youtu.be/715_nqYKpQE`

Company Record Management
`https://youtu.be/-qTE7oYgnmQ`

Certifier Record Management
https://youtu.be/0q6CxFrry70

Invoice Record Management
https://youtu.be/8OMWzQnqSdw

Notification Change Conversion Rate Request
https://youtu.be/4fmbX4s_JFw

7 Conclusion

This section refers to the last step of our research method, which is the Conclusion. Based on the outcome of the study I conclude that the best possible solution for auditing timber-to-charcoal process is to use a blockchain-based solution to limit trusted third-party organizations and get the immutable data for audit purposes. Hyperledger Fabric provides more features and functionality, focusing specifically on enterprises managing their complex assets with ease, providing confidentiality, Immutability, and highly scalable network growth. It uses conventional and most commonly used programming languages that can assist the developers in starting development without investing more time in learning.

I followed the Hyperledger Fabric and audit to the timber-to-charcoal business process as a case study and developed a blockchain solution. I followed the design science method and came up with a working blockchain application after Analyzing, designing, implementing, and testing the solution with compiled sample data.

I found the pros and cons of this technology after evaluating the results and outcome of the application. It provides a flexible access control structure with roles and authorities compared to other blockchain technologies because of its permission nature. Decentralization aid the audit process to perform quicker than a traditional business process, providing access to real-time, and guaranteed untempered data. It facilitates the audit process but is hard to implement and manage in terms of changes in requirements due to Command Line Interface (CLI) and scripting the configuration.

I tested this application with a self compiled sample data, but for future work, this study and software solution can help to map the real-world data and can provide results and insights to the readers to help in their research field work. However, this solution can be used to calculate and benchmark the performance of hyperledger fabric with the help of different tools.

References

[1] *A Blockchain Platform for the Enterprise¶*. URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/.

[2] Tagrid Alshalali, Kenneth M'Bale, and Darsana Josyula. "Security and Privacy of Electronic Health Records Sharing Using Hyperledger Fabric". In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2018, pp. 760–763.

[3] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. Porto, Portugal: Association for Computing Machinery, 2018. ISBN: 9781450355841. URL: https://doi.org/10.1145/3190508.3190538.

[4] Frederik J. Zuiderveen Borgesius and Wilfred Steenbruggen. "The Right to Communications Confidentiality in Europe: Protecting Privacy, Freedom of Expression, and Trust". In: *Theoretical Inquiries in Law* 20.1 (2019), pp. 291–322. URL: https://doi.org/10.1515/til-2019-0010.

[5] Pablo Garcia Bringas, Iker Pastor-López, and Giuseppe Psaila. "BlockChain Platforms in Financial Services: Current Perspective". In: *Business Systems Research Journal* 11.3 (2020), pp. 110–126. URL: https://doi.org/10.2478/bsrj-2020-0030.

[6] Christian Cachin et al. "Architecture of the hyperledger blockchain fabric". In: *Workshop on distributed cryptocurrencies and consensus ledgers*. Vol. 310. 4. Chicago, IL. 2016.

[7] Caixiang Fan et al. "Performance evaluation of blockchain systems: A systematic survey". In: *IEEE Access* 8 (2020), pp. 126927–126950.

[8] Kim Fenrich. "Securing your control system: the" CIA triad" is a widely used benchmark for evaluating information system security effectiveness". In: *Power Engineering* 112.2 (2008), pp. 44–49.

[9] Eshani Ghosh and Baisakhi Das. "A Study on the Issue of Blockchain's Energy Consumption". In: Jan. 2020, pp. 63–75.

[10] Alan R. Hevner et al. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), pp. 75–105. ISSN: 02767783. URL: http://www.jstor.org/stable/25148625.

[11] IBM. *What is Blockchain*. 2020. URL: https://www.ibm.com/ae-en/blockchain/what-is-blockchain?.

[12] Prince Waqas Khan, Yungcheol Byun, and Namje Park. "A Data Verification System for CCTV Surveillance Cameras Using Blockchain Technology in Smart Cities". In: *Electronics* 9 (Mar. 2020), p. 484.

[13] In Lee and Kyoochun Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises". In: *Business Horizons* 58.4 (2015), pp. 431–440. ISSN: 0007-6813. URL: https://www.sciencedirect.com/science/article/pii/S0007681315000373.

[14] Hongyu Li et al. "Blockchain-Based Data Preservation System for Medical Data". In: *Journal of Medical Systems* 42.8 (June 2018), p. 141. ISSN: 1573-689X. URL: https://doi.org/10.1007/s10916-018-0997-3.

[15] Zhiyong Li. "Will blockchain change the audit?" In: *China-USA Business Review* 16.6 (2017), pp. 294–298.

[16] Han Liu, Dezhi Han, and Dun Li. "Fabric-iot: A Blockchain-Based Access Control System in IoT". In: *IEEE Access* 8 (2020), pp. 18207–18218.

[17] Ning Lu et al. "A secure and scalable data integrity auditing scheme based on hyperledger fabric". In: *Computers & Security* 92 (2020), p. 101741.

[18] Shikha Maheshwari. *Blockchain basics: Hyperledger Fabric*. 2018. URL: https://developer.ibm.com/technologies/blockchain/articles/blockchain-basics-hyperledger-fabric/.

[19] Mariia Markovska. "Modelling Business Processes on a Blockchain Eco-System (BPMN)". In: 2019.

[20] Fredrik Milani et al. "Business Process Redesign Heuristics for Blockchain Solutions". In: Oct. 2020, pp. 209–216.

[21] Ritesh Modi. *Solidity programming essentials: a beginners guide to build smart contracts for Ethereum and blockchain*. Packt, 2018.

[22] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[23] Minh Quang Nguyen, Dumitrel Loghin, and Tien Tuan Anh Dinh. "Understanding the scalability of Hyperledger Fabric". In: *arXiv preprint arXiv:2107.09886* (2021).

[24] NguyenGiang-Truong and KimKyungbaek. "A Survey about Consensus Algorithms Used in Blockchain". In: *Journal of Information Processing Systems* 14.1 (Feb. 2018), pp. 101–128.

[25] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. eprint: https://doi.org/10.2753/MIS0742-1222240302. URL: https://doi.org/10.2753/MIS0742-1222240302.

[26] Bruno Rodrigues et al. “A blockchain-based architecture for collaborative DDoS mitigation with smart contracts”. In: *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, Cham. 2017, pp. 16–29.

[27] P Sajana, M Sindhu, and M Sethumadhavan. “On blockchain applications: hyperledger fabric and ethereum”. In: *International Journal of Pure and Applied Mathematics* 118.18 (2018), pp. 2965–2970.

[28] Omar S Saleh, Osman Ghazali, and Muhammad Ehsan Rana. “Blockchain based framework for educational certificates verification”. In: *Studies, Planning and Follow-up Directorate. Ministry of Higher Education and Scientific Research, Baghdad, Iraq. School* (2020).

[29] João Sousa, Alysson Bessani, and Marko Vukolic. “A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform”. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2018, pp. 51–58.

[30] K Stephens and MT Roszak. “A study of the role and benefits of third party auditing in Quality Management Systems”. In: *Journal of Achievements in Materials and Manufacturing Engineering* 43.2 (2010), pp. 774–781.

[31] Harish Sukhwani et al. “Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)”. In: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. 2018, pp. 1–8.

[32] Mueen Uddin. “Blockchain Medledger: Hyperledger fabric enabled drug traceability system for counterfeit drugs in pharmaceutical industry”. In: *International Journal of Pharmaceutics* 597 (2021), p. 120235. ISSN: 0378-5173. URL: https://www.sciencedirect.com/science/article/pii/S0378517321000399.

[33] Petros Wallden and Elham Kashefi. “Cyber security in the quantum era”. In: *Communications of the ACM* 62.4 (2019), pp. 120–120.

[34] Haiyan Wang and Jiawei Zhang. “Blockchain Based Data Integrity Verification for Large-Scale IoT Data”. In: *IEEE Access* 7 (2019), pp. 164996–165006.

[35] Shuai Wang et al. “An Overview of Smart Contract: Architecture, Applications, and Future Trends”. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 108–113.

[36] Wikipedia contributors. *Legality of bitcoin by country or territory — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Legality_of_bitcoin_by_country_or_territory&oldid=1054102785. [Online; accessed 10-November-2021]. 2021.

[37] Daniel Davis Wood. “ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER”. In: 2014.

[38] K. Wüst and A. Gervais. “Do you Need a Blockchain?” In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2018, pp. 45–54.

[39] Teng Xu, James B. Wendt, and Miodrag Potkonjak. “Security of IoT systems: Design challenges and opportunities”. In: *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2014, pp. 417–423.

[40] Wenbin Zhang et al. “A privacy-preserving voting protocol on blockchain”. In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE. 2018, pp. 401–408.

[41] Xiaoying Zheng, Yongxin Zhu, and Xueming Si. “A Survey on Challenges and Progresses in Blockchain Technologies: A Performance and Security Perspective”. In: *Applied Sciences* 9.22 (2019). ISSN: 2076-3417. DOI: 10.3390/app9224731. URL: https://www.mdpi.com/2076-3417/9/22/4731.

[42] Zibin Zheng et al. “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends”. In: June 2017.

Appendix

I. Acronyms

API Application Programming Interface. 45, 47, 52, 67, 77, 78, 81

BPMN Business Process Model Notation. 14, 25

CA Contract Accounts. 11, 12

CA Certificate Authority. 15, 46, 47, 51, 52, 73

CLI Command Line Interface. 47, 49, 58, 75, 77, 84

DDOS Distributed Denial of Service. 76

EOA Externally Owned Accounts. 11–13

ERD Entity Relationship Diagram. 22, 27, 33–35, 42, 80

ERP Enterprise Resource Planner. 16

IDE Integrated Development Environment. 44

IoT Internet of Things. 18, 19

MSP Membership Service Provider. 6, 15, 18, 49, 73, 81

NPM Node Package Manager. 45

PBFT Practival Byzantine Fault Tolerance. 10

POS Proof-of-Stake. 10

POW Proof-of-Work. 10

SDK Software Development Kit. 7, 51

TLS Transport Layer Security. 76

II. Glossary

Certificate Authority The Certificate Authority (CA) provides a number of certificate services to users of a Blockchain. More specifically, these services relate to user enrollment, transactions invoked on the Blockchain, and TLS-secured connections between users or components of the Blockchain. 15

node Blockchain nodes refer to a network's stakeholders and/or their devices, which are designated to keep a copy of the distributed ledger and serve as communication points that execute various essential network functions. 9, 15

World State The world state describes the state of the ledger at a given point in time. It's the database of the ledger. 14

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Muhammad Zubair**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   A Blockchain Solution for Auditing of Timber-to-Charcoal Process,

   supervised by Mubashar Iqbal and Dr. Fredrik Payman Milani.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Muhammad Zubair
11/11/2021