Thesis Title

A thesis submitted in fulfilment of the requirements
for the degree of Master of Technology/PhD

by

authorname

DEPARTMENT OF CIVIL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May 2019

Certificate

It is certified that the work contained in this thesis entitled ”Thesis Title” by ”author-name” has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

Thesis Supervisor

May 2019

Professor

Department of Civil Engineering

Indian Institute of Technology Kanpur

Abstract

Name of the student: **authorname**	Roll No: **Roll No.**
Degree for which submitted: **M.Tech.**	Department: **Civil Engineering**
Thesis title: **Thesis Title**	
Thesis supervisor: **Thesis Supervisor**	
Month and year of thesis submission: **May 2019**	

Enter Abstract Content here...

Acknowledgements

I would extend my sincerest gratitude...

Contents

List of Figures

List of Tables

Abbreviations

EVM	**E**therium **V**virtual **M**achine
PTE	**P**erformance **T**raffic **E**ngine
PKI	**P**ublic **K**ey **I**nfrastructure
RC	**R**einforced **C**oncrete

Symbols

D^{el}	elasticity tensor
σ	stress tensor
ε	strain tensor

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Motivation

In the current scenario, patient data is essential, and so is a patient's privacy. Doctors need data to examine patient efficiently, and many pharmaceutical companies need this data for their research. A patient needs this data, along with medical advice, so that he can make an informed decision about his health and life. He also needs to keep a record of his/her medical advises and treatments. He also needs this data so that he could talk to multiple experts and then make a judgment about his ongoing treatment. This scenario calls for the efficient online sharing of the patient data, such that it is quickly available for the desired party, but at the same time patient should be in control of his data, and who looks at his data. His privacy should not be compromised at any point. Also, there has been a continuous increase in fraud of supplying medical drugs to the patient without prescriptions. This misuse of medical drugs can become the reason for death by overdose.

1.1.1 Why patient data is needed?

For 21st century, data is the new oil. Healthcare sector is no exception, and it needs an efficient data collection mechanism. Analysis of these data helps doctors, for proactive interventions and thus providing better patient treatment and care. It is important for public agencies to collect these data so as to be able to make predictions about epidemic spread and then take concrete steps to curb it. But the relationship between patient and Doctors is unique and we must pass every information which Doctors, think is necessary for treatment, but Doctors should also respect patient privacy, and must not leak the sensitive information. With the rise in machine learning based inference tools, it is of utmost importance that patient data must be stored in a way that while it allows one to make a statistical inference, no information is leaked about any patient. Providers of health care services need information not only at the point of service but also at the point of decision making in a format that maximizes the decision-making process

1.1.2 Present methods of data collection

Currently, patient records are maintained manually or electronically. Manual patient records vary with the patient of the degree of accuracy, reliability, and accessibility. While some patient maintains their entire medical history, others do not. Electronic data collection is done in silos with various health care institutions and is a potential threat to a patient's privacy. While we do not judge the intentions of Health care providers, but a large scale cyber attack may leak private date of many patients.

1.1.3 Most Important Benefits of using blockchain in health sector

Following are the ways by which the blockchain can be used to increase security, privacy, and interoperability of patient's data in healthsector :

1. You can track who is accessing the data and when they accessed it
2. data can always be verified at any later point of time

One must also keep in mind that the only aim of patient data collection is not to improve the ease of statistical inference and provide reliable patient care, it should also improve the accessibility of the patient to various agents in health care ecosystem. It should bring together Doctors, pharmacy, insurance providers and patient. In this we aim to build such an ecosystem, that connects all the pillars, improves accessibility, does efficient data collection, but at the same time, gives utmost regards to patient privacy, and at no point patient data is susceptible to any cyber attack. We use Blockchain to achieve our objective and thus having distributed data storage, and hence avoiding any single point of failure.

1.1.4 Desirable outcome

We desire to have following setup for patient data sharing.

- A decentralized system which have all the patient's sensitive data like prescription dosage and bill amounts in encrypted format using patient's public key
- Blockchain, in which the Patients, Doctors/Labs, Insurance companies, pharmacies and other health-care institutions are peers.
- Any modification in the data (add/delete/update) could be done only through smart contracts having strict access policies.
- For any type of transaction endorsement is required from all the peers.
- All the access logs would be maintained on the Blockchain
- Anyone can ask for re-encryption key, but only patient can provide the re-encryption key to whoever he wants.

- Patient should be able to request appointment, pay bills, fetch prescription,lab reports, buy medicine, register for insurance
- Labs/Doctor should be able to generate Prescription, reports, bills and update it on the ledger. Doctors should be able to accept appointments.
- Insurance companies should be able to register patient for insurance plans, pay the bills for users with an automation system via smart contract
- Anyone should be able to ask for re-encryption key against a patient
- Only patient should be able to generate re-encryption key
- Prescription dosages and lab reports must be locked until the payment is made by the patients
- Patients should be able to fetch their complete medical history

Chapter 2

Blockchain Review

2.1 Blockchain Introduction

Blockchain is a shared, immutable database in a decentralized network. This immutable property of blockchain comes from the hash functions. Blockchain is a chain of blocks with each block containing previous block hash, current block hash, and some data. How the data is stored inside a block depends on the blockchain type. So, the bitcoin blockchain, for example, stores the details about a transaction in here, such as the sender, receiver, and the number of coins. Hash in the block can be compared to a fingerprint, which helps in identifying a block and all of its contents because it's always unique, just like a fingerprint. After the creation of the block, it's hash is calculated and changing something inside the block causes the mixture to change as well. So, if the fingerprint of a block changes, it no longer is the same block. The third element inside a block is the hash of the previous block, which provides integrity. So, replacing the content of any block will invalid all the following blocks. Using hash functions to prevent tampering is not enough. There can be three types of blockchain network :

- **Public blockchain** : In public blockchains, anyone can be a part of network with each node having their own ledger where the world state is decided by the longest chain holded by the majority. Example is bitcoin and etherium.
- **Private blockchain** : In private blockchain participants are already decided and only they can commit the transactions to the ledger
- **Permissioned blockchain** : This is a mix of both private and public blockchain, where participants are involved from both sectors. Hyperledger fabric is an example where the ledger being maintained by trusted 3rd party organization but the network can be designed where anyone can submit the transaction.

2.1.1 Terminologies

Assets : There are fundamentally two types of assets. There are tangible assets like house, laptop, phone etc. and then there are intangible assets like house loan, bonds, mutual funds etc. These intangible assets can be further categorized into financial assets like bonds and digital assets like songs.
Blockchain Networks : They are linkages between organization where transaction involving movement of assets and services occurs.
Ledger : It is a system of record for business networks
Consensus : It is a mutual agreement state for the validity of transaction reached by all of the blockchain network participants to ensure that they have the same copy of ledger
Digital Signature : It is a PKI method which uses the user's private key to generate a digitally coded string for the content of a document. This digital encoding can be used by anyone with the user's public key to verifying the integrity and ownership of the document

Chapter 3

Hyperledger Fabric

3.1 Introduction

It's an open source permissioned blockchain project first launched in year 2016. Hyperledger as an organization has now 10 active open source software projects. This project was launched in 2015. This helps in building permissioned blockchain network for business enterprises. It provides lot of capabilities that are both modular and pluggable, which is an important property because most of the blockchain frameworks are not exactly compatible to provide such feature. One of the example of pluggable service is the identity management where you can use a certificate authority of your choice. Also one of the biggest appeals of this type of framework is the segregation of the consensus and the chaincode execution which is unlike the etherium virtual machine mention in abbreviation(EVM) where the consensus is reached in the etherium network when the smart contract is run on each node's local machine in the entire network to provide the consensus. So, hyperledger fabric is more efficient model because both the consensus and execution of chaincode is happening in different spaces at the same time allowing for higher transactions throughput.

3.2 Nodes and roles

1. **MSP(Membership Service Provider)** : This node acts a certificate authority. Hyperledger fabric architecture uses x.509 certificates through public key infrastructure and each organization within a business network runs their won certificate authority.

2. **Client Node** : This node is used for the end user where the middleware code resides and a rest api server to interact with the middleware code.

3. **Peer** : Peer nodes are actually the backbone of the hyperledger fabric network. There are two types of peers :

   (a) Committing peer : updates the transaction in the ledger.

   (b) Endorsing peer : This node actually runs the smart contract to validate the transaction.

4. **Ordering Node** : These nodes runs the transactions, re-orders them if required and decides in which block the transactions will be included. These nodes do not have chaincodes installed on them and also does not hold the ledger. This node does not need to be present in each organization as opposite to the MSP nodes, but at least one of the organization has to run this.

3.3 More Terminologies :

1. **Genesis block** : This is the first block that is committed on the ledger. It has information on how to bootstrap the network, and also contains one big configuration block of certificates of all the organizations and peers in the network. So, this genesis block is downloaded primarily on the ordering service node.

3.4 7 Steps in submitting a transaction

1. **Transaction Proposal** : Channel name, chaincode name , chaincode version, function name which needs to be invoked for submitting a transaction or which needs to queries to get some results from the ledger, function arguments are gathered from client application. A transaction proposal is formed along with required ca, and orderer certificates. This transaction proposal is then submitted to the endorsers nodes.

2. **Transaction Proposal Execution** : Each node have endorsement policy already decided before the start of network. This endorsement policy can vary with the smart contracts. Once the transaction proposal is submitted by the client application, it will then require all the endorsement signatures from respective endorsing nodes decided in endorsement policy. While executing the transaction against the smart contract, endorser nodes checks things like if the user is allowed to do this transaction or not using access control policies.

3. **Transaction Proposal Response** : This transaction will be verified when these endorser nodes sends back the transaction and marking it as a good transaction along with signatures. The signature contains the read and write sets. Read sets contains information that this is the data read by endorser node and in write set will involve the data that endorsing nodes is writing in the database.

4. **Transaction order at ordering service** : Once the transaction is verified at each of the endorsers, the transaction is then submitted to the ordering service. These ordering services are getting many transactions at the same time from many other applications, so they performs the function to group them into block. Different consensus algorithms could be used here for ordering like solo orderer, kafka or zookeeper, raft.

5. **Transaction Delivery** : These blocks are then sent to each peer in the network.
6. **Transaction Validation** : The peers executes the transaction again in the block and checks if they executes correctly. They also validate the transaction against the endorsement policy. Also, Read and write sets are verified as well again from the transaction. Valid transactions are added to the world state while invalid transactions still remain on the ledger but are not updated in the world state.
7. **Transaction Notification** : The application then will be notified of the state of the transaction i.e, whether transaction was valid or not and thus confirming the application that valid transactions have been successfully added to the ledger.

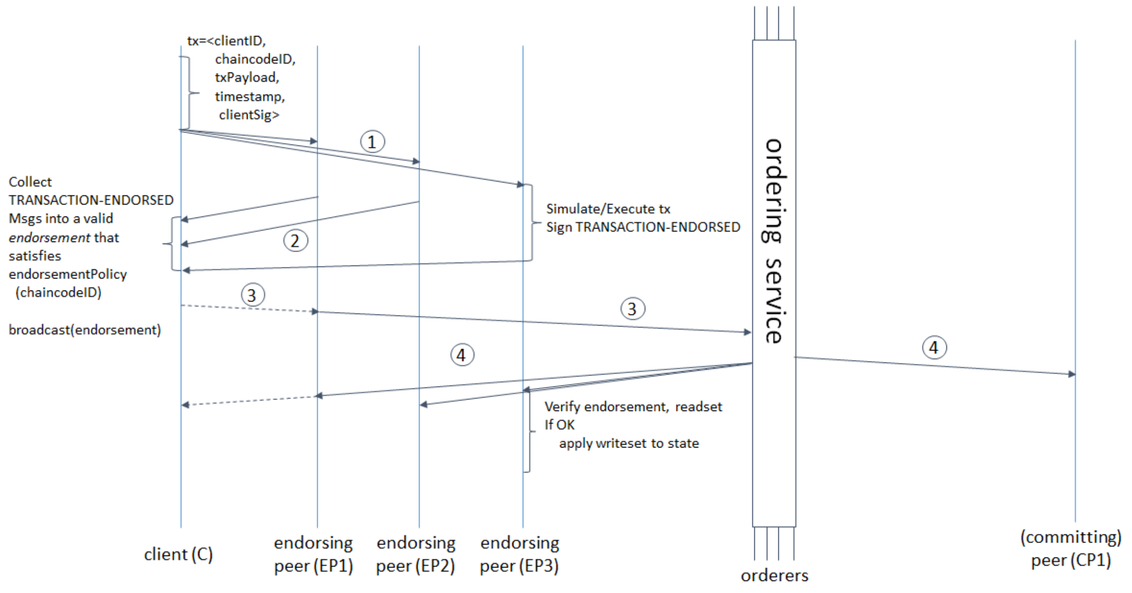

tx=<clientID, chaincodeID, txPayload, timestamp, clientSig>
1
Collect TRANSACTION-ENDORSED Msgs into a valid endorsement that satisfies endorsementPolicy (chaincodeID)
2
Simulate/Execute tx
Sign TRANSACTION-ENDORSED
3
broadcast(endorsement)
3
ordering service
4
4
Verify endorsement, readset
If OK
apply writeset to state
client (C)
endorsing peer (EP1)
endorsing peer (EP2)
endorsing peer (EP3)
orderers
(committing) peer (CP1)

Figure 3.1: Transaction flow
src: urlhttps://hyperledger-fabric.readthedocs.io/en/release-1.4/

Each organization in the network can run as many peers as they want but needs to have at least one peer and one ca node running on them. But, it's possible to have zero ordering service node in an organization. There is an option provided by the fabric network to use an external state database i.e, couchdb if developer does not choose leveldb. Couchdb is claimed to better when we have to perform rich queries but level db is more efficient when we do not have to do some extensive search in the database. LevelDB is the default state database provided by the hyperledger fabric framework. Each organization in the network runs these fabric nodes in docker containers which are actually little processes or little virtual machines with their own isolated environments.

3.5 Raft consensus algorithm

Every business network in modern world needs to rely on availability of systems to work. So, single machine doing all the job is out of option, simply because they can crash and needs maintenance, so downtime is definitely not zero. The solution that comes in mind is using multiple systems doing the same job in a distributed network. But in order to do that multiple problems still needs to solved first, for eg. what if the intercommunication between nodes is not reliable, what if they are prone to crashes. Also, for large business networks, it is not feasible to always have engineers doing the repairing job onsite. This is where raft consensus algorithm comes in scenario. The idea is to store the same data or commands in the logs. So, when one machine crashes then another machine can be activated in it's place and take up the work of crashed nodes. This helps in increasing the reliability of the architecture. So, raft is a consensus algorithm where replicated logs are managed. Any node in raft consensus can have only three states which are candidate, follower, or a leader as shown in 3.2

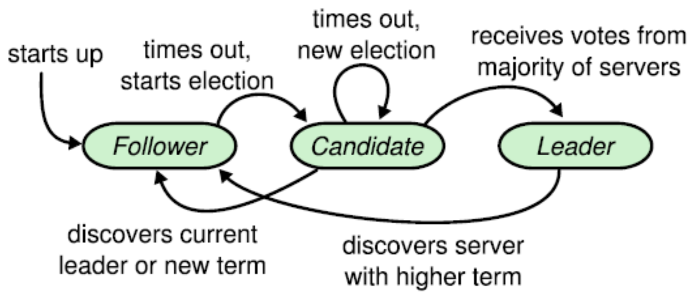

starts up
times out, starts election
times out, new election
receives votes from majority of servers
Follower
Candidate
Leader
discovers current leader or new term
discovers server with higher term

FIGURE 3.2: Node states in raft consensus
src: `https://medium.freecodecamp.org/`

All the client requests are handled by the leader, and followers perform only a passive role.This consensus involves the first election of a distinguished leader. The leader is then given the complete responsibility to maintain the replicated logs. The duty consists in taking log entries from clients followed by replication on other servers and telling the server when it is safe to apply log entries to their state machines. Logs management are simplified because logs are replicated without consultancy from other servers. When the leader node fails or become disconnected from the other servers, then a new leader is elected. Raft consensus algorithm can be divided into three independent components, which are :

1. **Election of a leader** : When an existing leader fails then a new leader is chosen. Raft leaders are elected after some arbitrary length of time interval. At the start of each interval one or more than one candidate tries to become leader. When a node wins the election with a clear majority then it is elected as the new leader. Situations where more than one node received the equal majority then none of them is elected as a leader and the election starts again for the new leader. To maintain the authority, periodic heartbeats are sent by the leader to all the followers using Remote Procedure Calls (RPCs). When the network is first started then all the nodes perform the role of followers. If no communication is received at the follower from the leader over an interval of time called the election timeout, then the leader node is assumed to be dead or disconnected and a new election starts.

2. **Replication of logs** : Log entries are accepted from the clients and are replicated across the distributed system, at the same time forcing other node log entries to synchronize. It is only up to the leader to decide that when is safe to execute commands in the log entries on a node. A new log entry is added only when the leader has replicated it to majority of nodes in the distributed system.

3. **Safety** : Raft ensures the safety property where if a particular log entry is applied by any server to its state machine then for the same log index, other servers cannot use a different command.

Chapter 4

Proxy Re-encryption Algorithm

4.1 Need for Re-encryption Scheme

Public key asymmetric key encryption system works very well when there is a need to transmit confidential messages over an insecure channel. However, such a scheme does not allow delegation of messages. E.g. While Alice could easily send message to Bob, but if Bob wants to forward that same message to Charles, he has to do decryption by his private key followed by encryption by Charles's public key. This leads to unwanted computational overload. One of the efficient ways through which Bob could delegate reading authority for some of his messages to Charles, without leaking information about his private key is through Proxy re-encryption scheme. In our work, we use the re-encryption algorithm proposed by Ateniese et al. [**?**] works on the assumption that Decisional Bilinear Diffie-Hellman is hard to decide.

4.2 Decisional Diffie-Hellman Assumption

For a cyclic group G of prime order q and generator g and group operator '.,' Decisional Diffie-Hellman assumption is that given three element of the group G as $x = g^a$, $y = g^b$, $z = g^c$ where $a, b, c \in \mathcal{Z}_q$ it is hard to decide whether $x.y = z$? Decisional Diffie-Hellman is based on the hardness of the discrete logarithm problem. For Bilinear group (which we define later) Decisional Diffie-Hellman assumption is that given $x = g^a,\ y = g^b,\ z = g^c,\ Q \in G$, where $a, b, c \in \mathcal{Z}_q$ it is hard to decide whether $Q = e(g, g)^{abc}$. Along with this we also assume hardness of Discrete logarithm problem.

4.3 Bilinear Groups

Bilinear Map: We say a map $e : G_1 \times \hat{G}_1 \rightarrow G_2$ is bilinear if

- $G_1,\ G_2$ are groups of same prime order q

- for all $a, b \in Z_q,\ g \in G_1$ and $h \in \hat{G}_1$ then $e(g^a, h^b) = e(g, h)^{ab}$
- map is non-degenerate i.e if g generates G_1 and h generates $\hat{G}_1$ then $e(g, h)$ generates G_2
- there exists a computable isomorphism form $\hat{G}_1$ to G_1

4.4 The Re-encryption Algorithm

While choosing a Re-encryption algorithm, few necessities are to be kept in mind.

- Not every message needs to be forwarded to another, and hence when a patient desire that only his doctor should be able to read a message, and should not be able to do that, without either doing decryption-encryption cycle or leaking his private key, he should be able to do that
- Patient should be able to choose, which messages to be forwarded, and which are not
- even though a patient may allow the doctor to forward his message (patient's data) to another doctor, that should not give doctor right to allow another doctor to be able to forward the same message (patient's data)
- delegation should be computationally inexpensive, and minimum changed must be done in the ciphertext

Based on our needs, we use the re-encryption algorithm suggested by Ateniese et. al.[**?**]. We provide a brief description of the algorithm below. Here **Type 1** encryption is such that the ciphertext could be decrypted only by the intended recipient and could not be forwarded and delegated. **Type 2** encryption could be decrypted by the recipient and could also be delegated and forwarded. During delegation, a new ciphertext is generated by a small modification of the original ciphertext.

- **Key generation:** public key of user is of form $pk_a = (Z^{a_1}, g^{a_2})$ and Secret key is $sk_a = (a_1, a_2)$. $Z^a = e(g, g)^a$
- Re-encryption key is: $rk_{A \to B} = g^{a_1 b_2}$
- **Type 1 Encryption**:
  - **Encryption:** $c_{a,1} = (Z^{a_1 k},\ m \oplus Z^k)$ where k is any random element of the group.
  - **Decryption:** Receiver knows a_1 as his first private key and can calculate a_1^{-1}, and hence can calculate Z^k from $Z^{a_1 k} = Z^{a_1 k / a_1} = e(g, g)^{a_1 k / a_1}$ message is reconstructed as $m \oplus Z^k \oplus Z^k$
- **Type 2 encryption :**

- **Encryption:** $c_{a,r} = (g^k, m \oplus Z^{a_1 k})$
- **Re-Encryption:** Compute $e(g^k, g^{b_2 a_1}) = Z^{b_2 a_1 k}$. Now the re-encrypted cipher text $c_{b,r} = (Z^{b_2 a_1 k}, m \oplus Z^{a_1 k})$
- **Decryption:**
  By 1st receiver: receiver can get $Z^{a_1 k}$ as $e(g^k, g^{a_1})$ and hence can get plain text as $m \oplus Z^{a_1 k}$
  By 2nd receiver (Charles): since Charles knows b_2 so Charles can get Z^{1/b_2} from which he can get $Z^{b_2 a_1 k}/Z^{b_2} = Z^{a_1 k}$ and hence can get m as $m \oplus Z^{a_1 k} \oplus Z^{a_1 k}$.

Chapter 5

Design and implementation

5.1 Design

The architecture is designed to achieve the following objectives:

1. Appointment requests, lab checkup reports, and dosage prescription must be committed to the ledger
2. Patient should be able to fetch his medical records
3. Sensitive data like dosage details, lab reports in prescription and amounts in bills must be encrypted with the patient's public key. So, the only patient should have access to his medical records
4. Patient should be able to allow anyone to see his medical records
5. A mechanism to capture all the bills and payment flow for lock/unlock of prescription dosage and lab reports
6. To involve insurance company in the architecture for automatic payments of bills
7. Patient's approval on bills for verification by the insurance company
8. Patient's should be able to register for an insurance plan with the insurance company.
9. Pharmacies should be able to use the prescription from the ledger to give medicines to user
10. Strict access policies ensuring assets are fetched by the rightful owner

Participants and Organizations

S.No.	Participants Involved	Organization
1.	Patients	PatientORG
2.	Doctors	HospitalORG
3.	Nurses	HospitalORG
4.	Labs	HospitalORG
5.	Insurance Companies	InsuranceORG
6.	Pharmacies	PharmacyORG

Table 5.1: Participants and their organizations

Main Assets

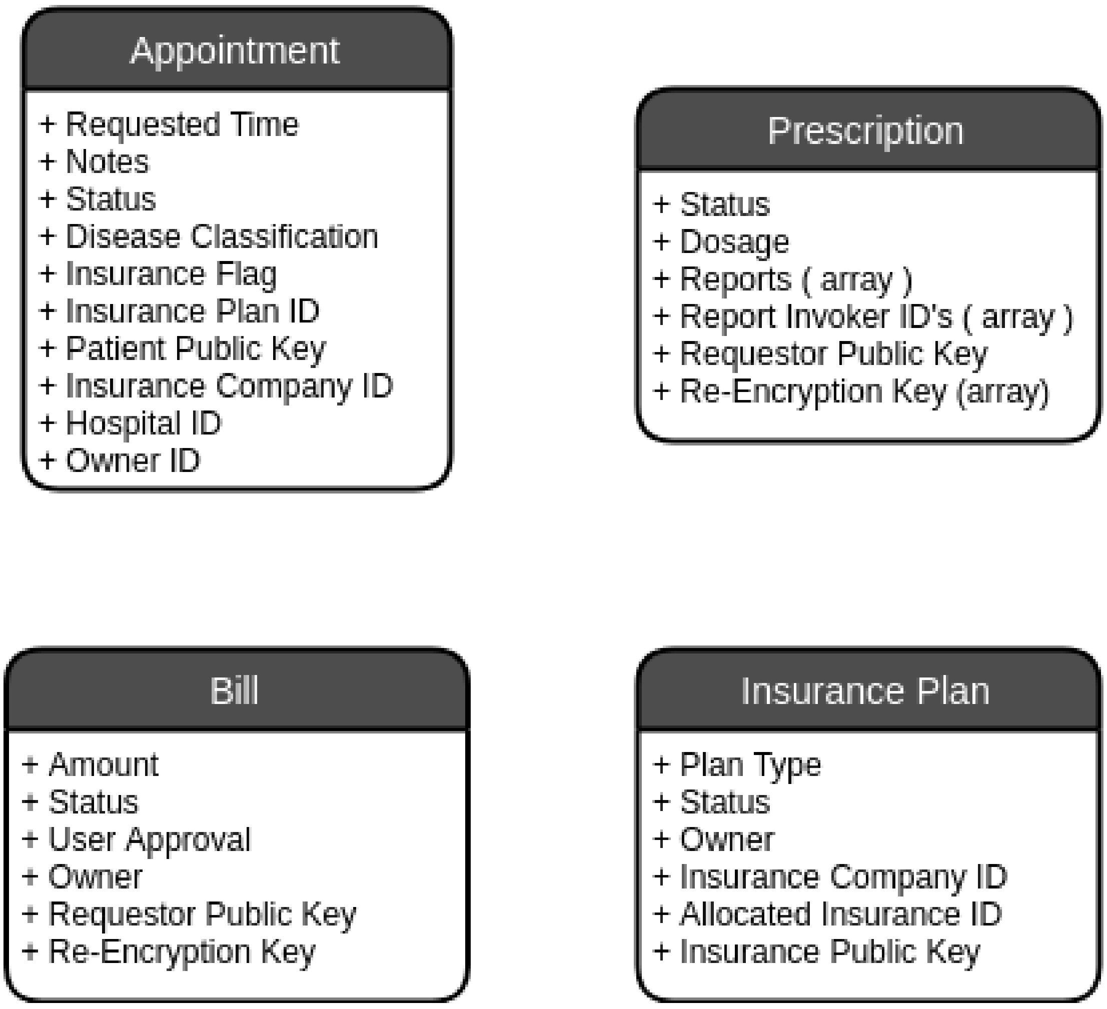


Figure 5.1: Assets

5.2 Architecture

5.2.1 Scenarios

There can be two possibilities either the patients have registered for health insurance, and the insurance company pays a bill for them, or the patient himself pays the bill. An important assumption in the architecture is that all the money transfer are being done outside the hyperledger fabric network. It's only the bill and other metadata details that are being updated on the ledger. This assumption was taken because all the bills amount that is generated against the patient have their amount encrypted with patient's public key. So, keeping a wallet in the network was not a good idea.

Case I: Patient with no health Insurance

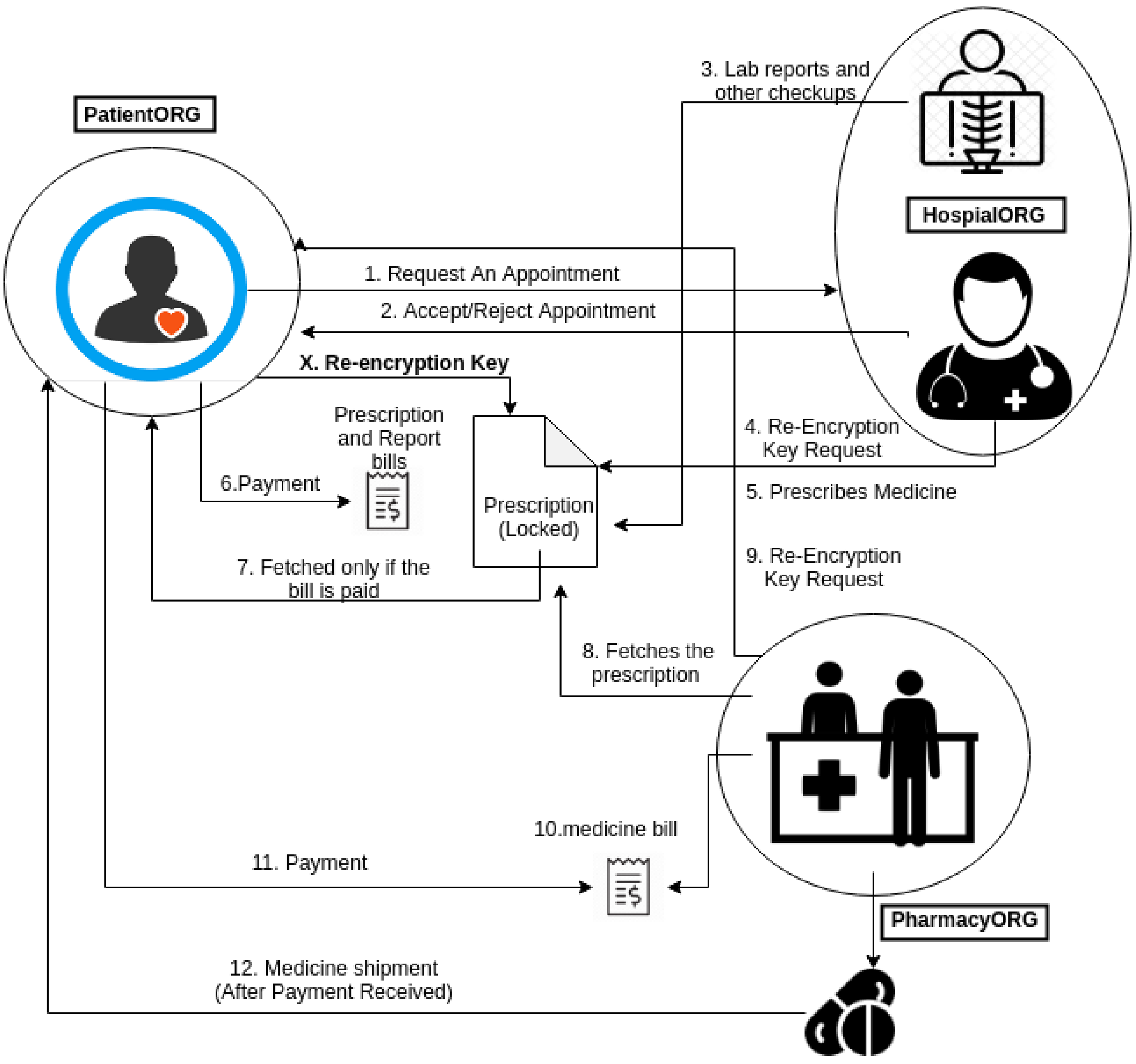

PatientORG
3. Lab reports and other checkups
HospialORG
1. Request An Appointment
2. Accept/Reject Appointment
X. Re-encryption Key
Prescription and Report bills
6.Payment
Prescription (Locked)
4. Re-Encryption Key Request
5. Prescribes Medicine
9. Re-Encryption Key Request
7. Fetched only if the bill is paid
8. Fetches the prescription
10.medicine bill
11. Payment
PharmacyORG
12. Medicine shipment (After Payment Received)

Figure 5.2: Without Health Insurance

1. Patient requests an appointment with a doctor for a particular time using a randomly generated appointment id which acts as a token for patient till he gets the medicine.

2. Response by the doctor to accept or reject the appointment depending on his schedule

   When the patient reaches the hospital, the doctor generates an empty prescription asset corresponding to the appointment token and adds it to the ledger. Now, the doctor can ask the patient to get some checkups done before getting a prescription.

3. Patient gives his token to Labs and encrypted lab reports with patient public key are added to the same prescription asset

   Whenever a lab report is added to the prescription, a corresponding bill is also generated.

4. To see the lab reports doctor asks the patient to update re-encryption key (**step X**) in prescription

5. Doctors updates prescription asset with dosage detail and also a bill is generated corresponding to that prescription

   when updating the prescription with dosage details the prescription is marked as 'COMPLETE'

6. Patient pays for the lab reports and prescription bill

   Whenever a patient pays for a bill, he marks that bill as 'PAYMENT SENT'. When the corresponding generator of that bill successfully verifies his account with the money transfer then he marks the bill as 'PAYMENT RECEIVED'. When the patient pays for all the generated bills then the prescription is marked as 'PAID', which unlocks the prescription.

7. After the payment for all bills, the patient can now fetch the prescription

8. Patient goes to the pharmacy and gives them their appointment token, with this appointment token pharmacy, fetches the prescription dosage

9. Pharmacy asks the patient to update the re-encryption key (**step X**)

10. Bill is generated by the pharmacy

11. Payment is made by the patient for bill generated by the pharmacy

12. Once the payment is confirmed then the pharmacy will ship the medicine

Case II: Patient with health Insurance

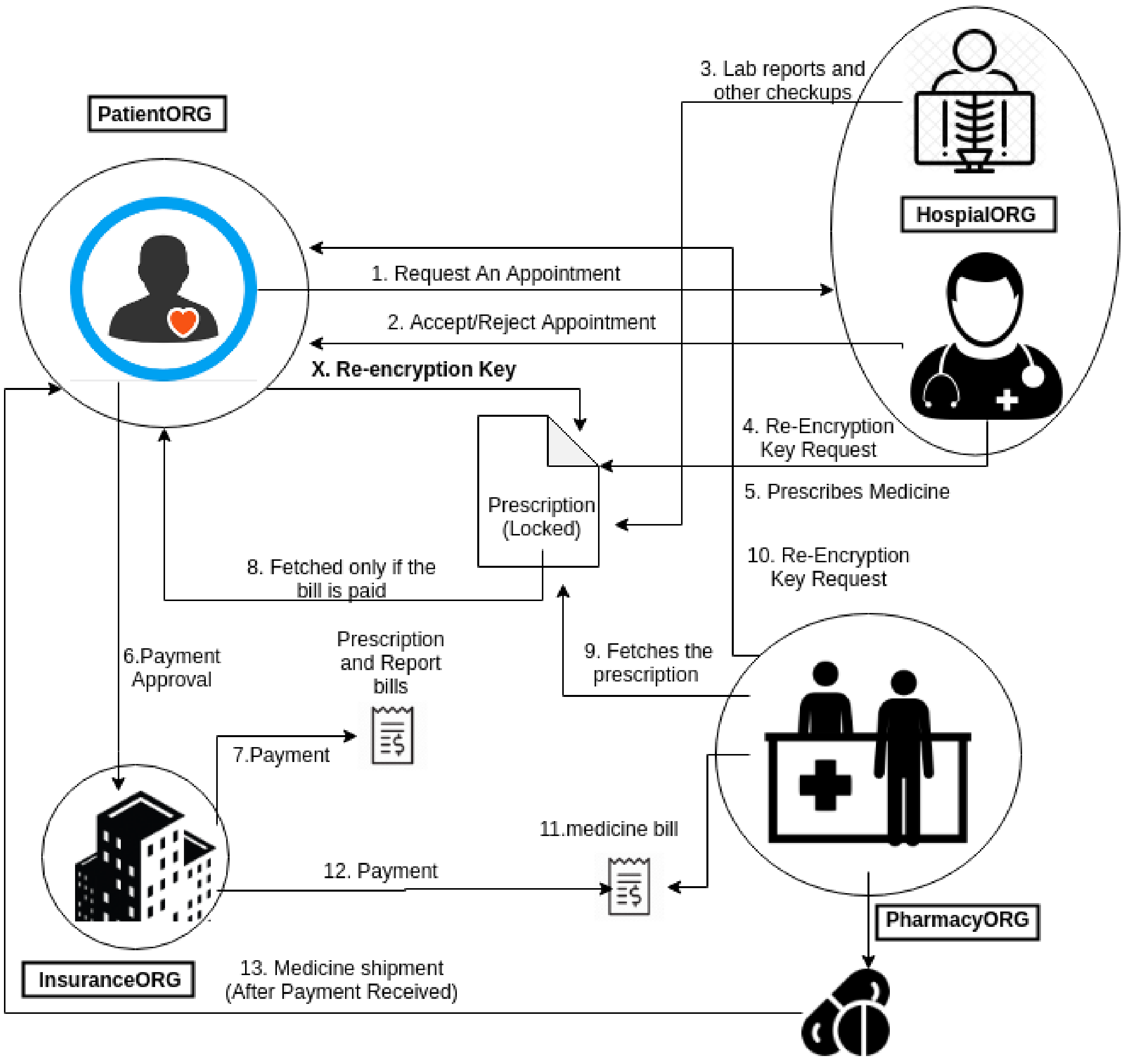

PatientORG
3. Lab reports and other checkups
HospialORG
1. Request An Appointment
2. Accept/Reject Appointment
X. Re-encryption Key
4. Re-Encryption Key Request
5. Prescribes Medicine
Prescription (Locked)
8. Fetched only if the bill is paid
10. Re-Encryption Key Request
6.Payment Approval
Prescription and Report bills
9. Fetches the prescription
7.Payment
11.medicine bill
12. Payment
PharmacyORG
InsuranceORG
13. Medicine shipment (After Payment Received)

FIGURE 5.3: Without Health Insurance

In case II 5.3 when the patient has opted for the insurance plan then he can provide his insurance id and insurance company id with at the time of appointment. At any further stages, payment of bills will be done automatically by the insurance company and the main differences from the case I are as follows :

6 After generation of bills by Labs, doctors or pharmacies, the patient will provide user approval on the bills by marking the bill as *APPROVED*.

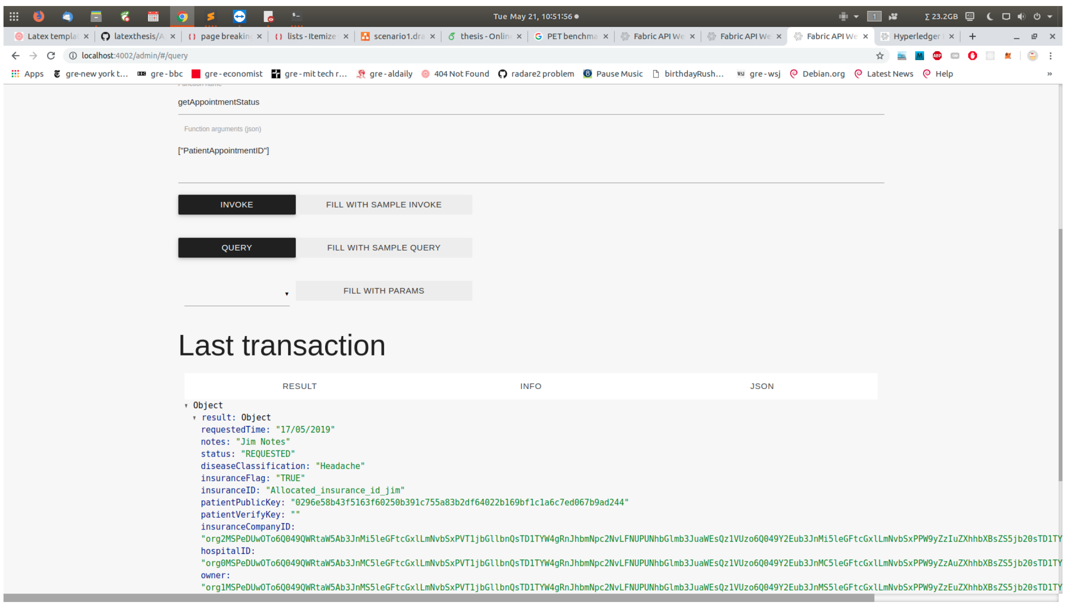
While providing the approval, the patient will also provide a re-encryption key for the insurance company which will be updated in the corresponding bill. When the patient provides an approval to the bill, payment requests for that bill are automatically added via smart contract to the corresponding insurance company's payment requests database.

7 Insurance company will verify the approval status and the owner of the bill followed by the actual payment.

When the insurance company pays for some bill then it also marks it as 'PAYMENT SENT' similar to the previous CASE I 5.2. Rest of the steps are the same as CASE I.

5.2.2 Logical View

The application provides a rest API interface using express JS server where the users can invoke the smart contracts as shown in 5.4 where a user is checking his appointment status using the appointment token and the smart contract 'getApppointmentStatus'.



getAppointmentStatus

Function arguments (json)

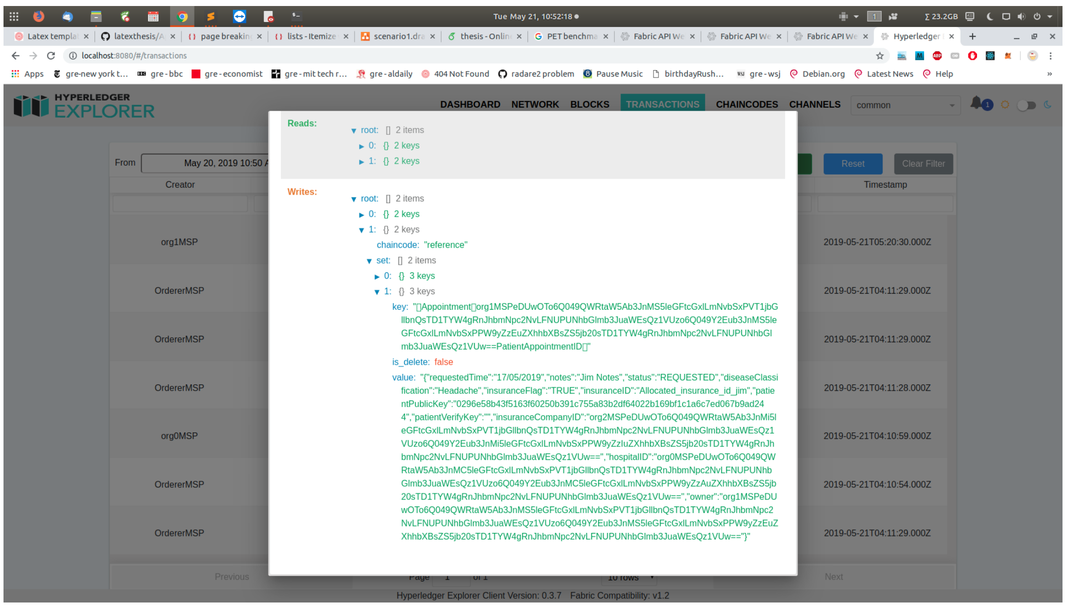
["PatientAppointmentID"]

Last transaction

```
▾ Object
  ▾ result: Object
    requestedTime: "17/05/2019"
    notes: "Jim Notes"
    status: "REQUESTED"
    diseaseClassification: "Headache"
    insuranceFlag: "TRUE"
    insuranceID: "Allocated_insurance_id_jim"
    patientPublicKey: "0296e58b43f5163f60250b391c755a83b2df64022b169bf1c1a6c7ed067b9ad244"
    patientVerifyKey: ""
    insuranceCompanyID:
    "org2MSPeDUwOTo6Q049QWRtaW5Ab3JnMi5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMi5leGFtcGxlLmNvbSxPPW9yZzIuZXhhbXBsZS5jb20sTD1TY
    hospitalID:
    "org0MSPeDUwOTo6Q049QWRtaW5Ab3JnMC5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMC5leGFtcGxlLmNvbSxPPW9yZzAuZXhhbXBsZS5jb20sTD1TY
    owner:
    "org1MSPeDUwOTo6Q049QWRtaW5Ab3JnMS5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMS5leGFtcGxlLmNvbSxPPW9yZzEuZXhhbXBsZS5jb20sTD1TY
```

FIGURE 5.4: Express JS frontend

Furthermore, there is an admin dashboard to analyze the network, and it's transactions which are powered by hyperledger explorer and is shown in 5.5



From May 20, 2019 10:50 A

Creator	Timestamp
org1MSP	2019-05-21T05:20:30.000Z
OrdererMSP	2019-05-21T04:11:29.000Z
OrdererMSP	2019-05-21T04:11:29.000Z
OrdererMSP	2019-05-21T04:11:28.000Z
org0MSP	2019-05-21T04:10:59.000Z
OrdererMSP	2019-05-21T04:10:54.000Z
OrdererMSP	2019-05-21T04:11:29.000Z

Page 1 of 1

```
Reads:
    ▾ root: [] 2 items
      ▸ 0: {} 2 keys
      ▸ 1: {} 2 keys

Writes:
    ▾ root: [] 2 items
      ▸ 0: {} 2 keys
      ▾ 1: {} 2 keys
          chaincode: "reference"
        ▾ set: [] 2 items
          ▸ 0: {} 3 keys
          ▾ 1: {} 3 keys
              key: "Appointmentorg1MSPeDUwOTo6Q049QWRtaW5Ab3JnMS5leGFtcGxlLmNvbSxPVT1jbG
                llbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMS5le
                GFtcGxlLmNvbSxPPW9yZzEuZXhhbXBsZS5jb20sTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGl
                mb3JuaWEsQz1VUw==PatientAppointmentID"
              is_delete: false
              value: "{"requestedTime":"17/05/2019","notes":"Jim Notes","status":"REQUESTED","diseaseClassi
                fication":"Headache","insuranceFlag":"TRUE","insuranceID":"Allocated_insurance_id_jim","patie
                ntPublicKey":"0296e58b43f5163f60250b391c755a83b2df64022b169bf1c1a6c7ed067b9ad24
                4","patientVerifyKey":"","insuranceCompanyID":"org2MSPeDUwOTo6Q049QWRtaW5Ab3JnMi5l
                eGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1
                VUzo6Q049Y2Eub3JnMi5leGFtcGxlLmNvbSxPPW9yZzIuZXhhbXBsZS5jb20sTD1TYW4gRnJh
                bmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUw==","hospitalID":"org0MSPeDUwOTo6Q049QW
                RtaW5Ab3JnMC5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhb
                Glmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMC5leGFtcGxlLmNvbSxPPW9yZzAuZXhhbXBsZS5jb
                20sTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUw==","owner":"org1MSPeDU
                wOTo6Q049QWRtaW5Ab3JnMS5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2
                NvLFNUPUNhbGlmb3JuaWEsQz1VUzo6Q049Y2Eub3JnMS5leGFtcGxlLmNvbSxPPW9yZzEuZ
                XhhbXBsZS5jb20sTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUw=="}"
```

Hyperledger Explorer Client Version: 0.3.7 Fabric Compatibility: v1.2

FIGURE 5.5: Caption

S.No.	**Smart Contract**	**Access Policy**	**Functionality**
1	requestAppointment	Patient	Appointment request with a doctor
2	getAllAppointment	Doctor	Fetch all appointment requests for a particular day
3	acceptAppointment	Doctor	Appointment acceptance
4	getAppointmentStatus	Patient	Appointment Request Status Check
5	generatePrescription	Doctor	Generation of empty prescription by the doctor when patient arrives
6	generateBill- -AndUpdate- -PrescriptionOthers	Labs	Addition of lab reports in prescription and Bill generation with given amount
7	generateBill- -AndUpdate- -PrescriptionDoctor	Doctor	Addition of Dosage in in prescription and Bill generation with given amount
8	getPending- -PaymentRequests	Patient	get all pending payment requests for an appointment
9	registerInsurance	Patient	Insurance Registration Request
10	getInsurance- -RegistrationRequests	Insurance Company	Fetch all registration requests for a particular day
11	registerInsurance- -RequestStatus	Patient	Insurance Registration Request Status Check
12	acceptInsurance- -RegistrationRequest	Insurance Company	Insurance Registration Request Acceptance
13	getBillStatus	Labs/Doctor	Bill Payment status check by the creator
14	getBillStatusPatient	Patient	Bill payment status check by the patient for an appointment
15	approvePaymentRequest	Patient	Approval by Patient for Bill Payment by Insurance
16	getBillStatusInsurance	Insurance Company	Fetches bill
17	getInsurance- -PaymentRequests	Insurance Company	Fetch all bill for the particular date
18	payInsurance- -PaymentRequests	Insurance Company	Payment of Bill by Insurance company for Patient
19	markPrescriptionPaid	Doctor	Prescription marked as paid
20	getPrescriptionPatient	Patient	Fetches the prescription
21	requestPrescriptionOthers	Anyone	Request for re-encryption key
22	allowPrescriptionOthers	Patient	Adds re-encryption key in the prescription
23	getPrescriptionOthers	Anyone	Fetches the prescription after all payments are made
24	getOrgUsers	Anyone	Fetches Users Unique Id’s
25	registerUserFabric	Anyone	Registers Unique Id’s

TABLE 5.2: Smart Contracts used in architecture

5.2.3 Process View

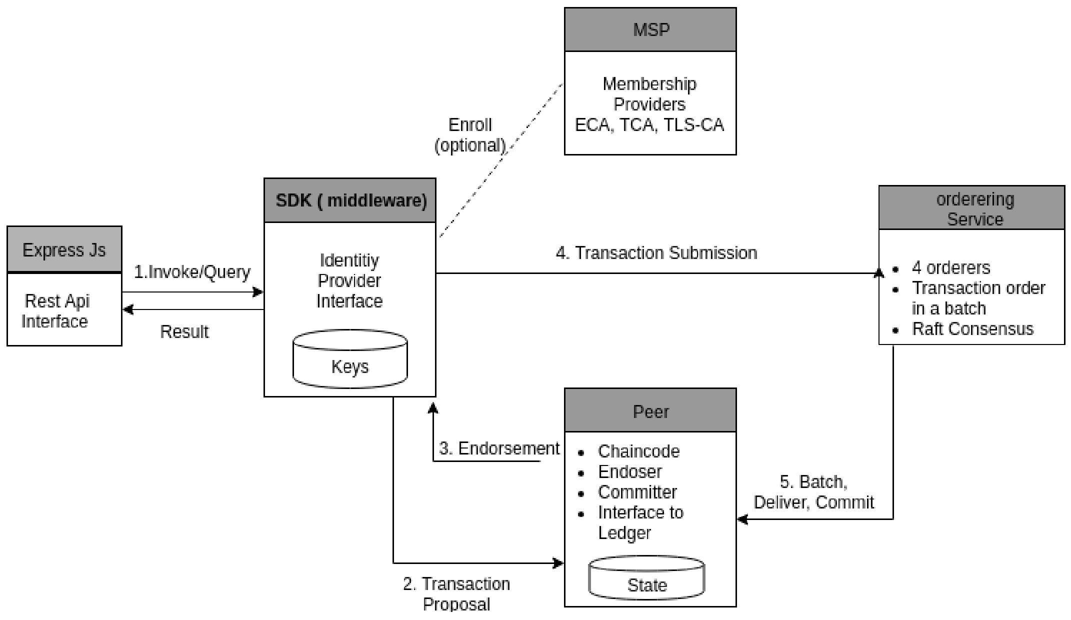


FIGURE 5.6: Process View

Express js server exposes rest API to the end user for submitting transaction proposal. These transactions are intercepted at the middleware node where cryptographic material of the invoker identity is added. This cryptographic material includes unique id assigned to the user who is invoking the transaction and also the unique organization id to which the user belongs. Also, this is the stage where the x.509 certificate is attached to the transaction proposal. If this user was not enrolled earlier then the middleware will interact with MSP to register it with the fabric network and a secret key is returned to the user. Once the transaction proposal is formed with information of the organization name, channel name,chaincode id, smart contract name and arguments then this transaction is then submitted to the committing peer of that organization. The peer then validates the transaction by executing it and at the same time, it also sends the transaction proposal to other endorsers according to the endorsement policy. Other endorsers also validate the transaction by executing it and prepares a Read set and write set. Read set contains the information that what did the endorsers read from database and write set includes the information that what did the endorsers write into the database. Once the read set and write set are prepared and attached to the proposal response then the endorsers attaches a signature in proposal response saying that it is a good transaction. After getting a good response from all the endorsers, the transaction is committed to the ledger by the peer but not update in the world state and then it is the submitted to the ordering service in step 4 in 5.6. The architecture is using raft as the consensus protocol, so if the orderer to which the transaction is submitted is not the leader, it submits it to the leader first then it is the leader's responsibility to assign one of the orderers and sends the transaction to it. The orderer then with many other transactions that came within that second orders it into a block whose maximum number of transaction capacity is 100. The size of the block, i.e., 100 transactions per block is already decided before the bootstrapping of the network.

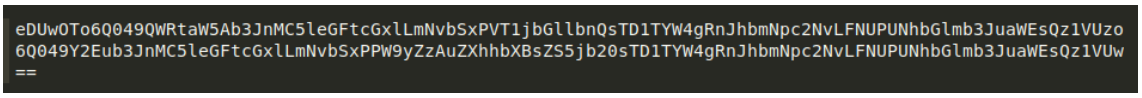
This block is then sent to each peer in the network. Transactions in the block are executed again for verification and the signatures are verified. Valid transactions are added to the world state. The application will then be notified of the state of the transaction i.e., whether it was a valid transaction or not.

5.2.4 Development View

The complete code of the architecture can be broken into the following modules :

1. Network setup module: This consists of docker and docker compose files having information related to the network like the number of organizations, number of peers in each organization, the volume that will be mounted in the docker containers, path to certificates and keys.
2. Network deployment module: This mainly comprises of ansible playback modules which are used to deploy the network. This module starts the runs the docker containers on each server. The modules specify that after the generation of cryptographic material at a single node which files will be sent to which host. Executes commands on each server to create a channel, joins peer on the channel according to the configuration, then installs the chaincode and instantiate it as well.
3. Business logic module: This comprises the actual logic of architecture, i.e., the chaincode where the 25 smart contracts were written ??. Each smart has its access policy implemented in it.

   Each user in the fabric network is assigned an id that is unique within an organization. This id looks something like 5.7



```
eDUwOTo6Q049QWRtaW5Ab3JnMC5leGFtcGxlLmNvbSxPVT1jbGllbnQsTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUzo
6Q049Y2Eub3JnMC5leGFtcGxlLmNvbSxPPW9yZzAuZXhhbXBsZS5jb20sTD1TYW4gRnJhbmNpc2NvLFNUPUNhbGlmb3JuaWEsQz1VUw
==
```

FIGURE 5.7: User id

This unique id does not need to be unique across the organization. Also, each organization is assigned an id which is unique across the network. So, in my architecture org0 is assigned the unique id 'org0MSP'. When the organization id is attached to the id in 5.7 then it becomes a unique id across the complete network. This unique id's can be used to implement access policies within smart contracts. E.g. in requestAppointment smart contract ?? this unique id along with the appointment id which was generated randomly by the application acts as a key for the asset appointment. Now when the patient tries to check his appointment status in the getAppointmentStatus smart contract then he will use the token which contains this unique id + random appointment id. In smart contract shim library is used to fetch the organization id and the user's id attached in the transaction proposal at the middleware. Then this id's are validated against the token submitted by the patient. If it's correct patient who requested Appointment for the given token then he will be able to check the status of the appointment otherwise a response saying 'You are not the correct owner of this appointment' will be returned to the client application.

4. Middleware module: Comprises of nodejs files, responsible to transaction proposal for creating a channel, chaincode installation, chaincode instantiation, chaincode invoke, chaincode query and user enrollment.

5. Application module: Express js server file containing (Jawa based token) JWT authentication of the user. Contains code to intercept the user's request and do the required encryption for some particular functions. For eg. In smart contract 'generateBillAndUpdatePrescription', the prescription dosage details, bill amount are encrypted using Patient's public key and only after that it is submitted to the middleware. Encryption occurs by spawning a synchronous thread on the server executing a python file which uses the umbral library for the encryption.

6. Testing module: Since the architecture is pretty big, so python scripts are written having curl requests with various test cases to test the complete chaincode in one go.

5.2.5 Physical View

The architecture is deployed over four servers running on the private network of IITK each having the following specifications :

- Operating System : ubuntu 18.04 LTS
- RAM : 16 GB
- Processor : Intel® Core^{TM} i7-4710HQ CPU 3.3GHz
- Cores : 4

Server0 is denoted as root orderer because the cryptographic node is generated at this node and then the keys and certificates are transferred to the correct server. Ansible playbook has been used to deploy the network since the policies inside ansible playbook can be defined with full control during deployment.

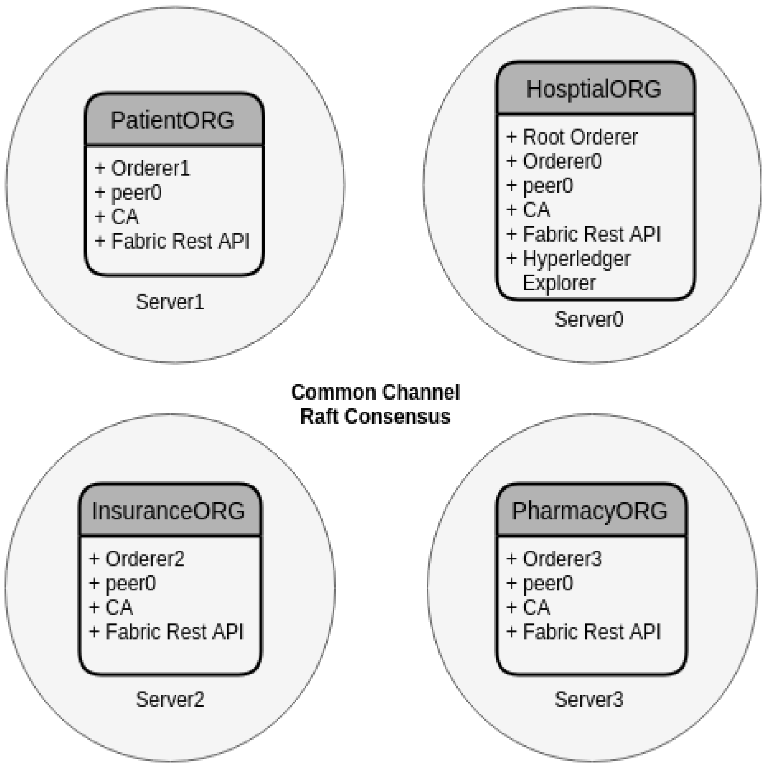


Figure 5.8: Physical View

Following were the docker containers running after the deployment on each server with the mentioned IP address in the table. Since the TLS was enabled in the network, grpcs connection protocol was used for intercommunication between containers on a different host.

S.No.	Node Type	IP Address	Port
1	Committing Peer	172.24.144.139	7051
2	Endorsing Peer	172.24.144.139	7053
3	Orderer	172.24.144.139	7050
4	ca	172.24.144.139	7054
5	Rest API (HospitalORG)	172.24.144.139	4000
6	Hyperledger Explorer	172.24.144.139	8080
7	Committing Peer	172.28.57.54	7051
8	Endorsing Peer	172.28.57.54	7053
9	Orderer	172.28.57.54	7050
10	ca	172.28.57.54	7054
11	Rest API (PatientORG)	172.28.57.54	4000
12	Committing Peer	172.27.27.175	7051
13	Endorsing Peer	172.27.27.175	7053
14	Orderer	172.27.27.175	7050
15	ca	172.27.27.175	7054
16	Rest API (InsuranceORG)	172.27.27.175	4000
17	Committing Peer	172.27.28.58	7051
18	Endorsing Peer	172.27.28.58	7053
19	Orderer	172.27.28.58	7050
20	ca	172.27.28.58	7054
21	Rest API (PharmacyORG)	172.27.28.58	4000

Table 5.3: Smart Contracts used in architecture

Chapter 6

Network Benchmark

To analyze the performance of the network, throughput, and latency's at various stages must be obtained while simulating the real usage scenario. This was done by using the Benchmark tool, Performance Trafic Engine (PTE) [pte] provided by the hyperledger community.

6.1 Methodology

Following are some definitions :

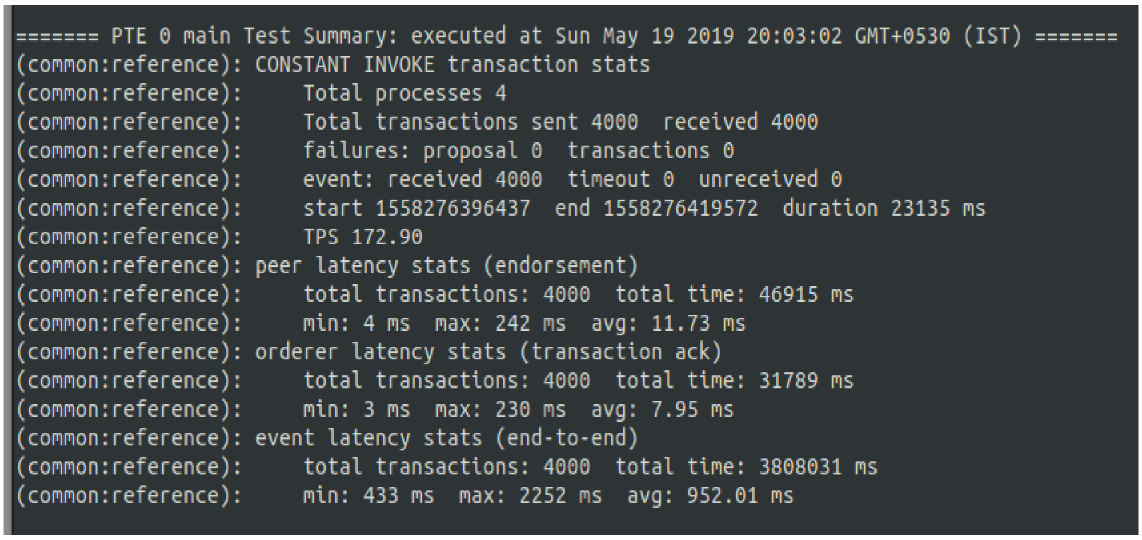
1. **Throughput** : the rate at which the transactions are committed to the ledger. Its measurement unit is transactions per second, i.e., tps
2. **Latency** : It's the intermediate time between the transaction proposal generation at the application end and committing of that transaction in the ledger. There can be many latencies in the network, and a few of the important ones under the analysis metrics are the following :
    (a) **Peer Latency** : Latency in intercommunication of peers while endorsement
    (b) **Orderer Latency** : Time taken at orderers while making batches of transactions
    (c) **Event Latency** : This is overall latency from the transaction proposal generation from the user till the user received back the response against the transaction

6.2 Experimental Setup

The setup was the same as the physical view 5.8 with PTE residing on server 0. Following are the configuration used for the benchmark :

1. Number of organizations: 4
2. Number of Peers per organization : 1
3. Number of Orderers : 4
4. Ordering consensus : Raft
5. Number of Processes : 4
6. Number of Process per Organization : 1
7. Total Transactions per process : 1000
8. Burst Frequency : [500, 200, 300, 100]
9. Burst Duration : [3000, 2000, 3000, 1000] (in ms)
10. Transaction frequency : constant

6.3 Output



```
======= PTE 0 main Test Summary: executed at Sun May 19 2019 20:03:02 GMT+0530 (IST) =======
(common:reference): CONSTANT INVOKE transaction stats
(common:reference):     Total processes 4
(common:reference):     Total transactions sent 4000  received 4000
(common:reference):     failures: proposal 0  transactions 0
(common:reference):     event: received 4000  timeout 0  unreceived 0
(common:reference):     start 1558276396437  end 1558276419572  duration 23135 ms
(common:reference):     TPS 172.90
(common:reference): peer latency stats (endorsement)
(common:reference):     total transactions: 4000  total time: 46915 ms
(common:reference):     min: 4 ms  max: 242 ms  avg: 11.73 ms
(common:reference): orderer latency stats (transaction ack)
(common:reference):     total transactions: 4000  total time: 31789 ms
(common:reference):     min: 3 ms  max: 230 ms  avg: 7.95 ms
(common:reference): event latency stats (end-to-end)
(common:reference):     total transactions: 4000  total time: 3808031 ms
(common:reference):     min: 433 ms  max: 2252 ms  avg: 952.01 ms
```

FIGURE 6.1: Benchmark Results

Chapter 7

Conclusions and Future Scopes

Privacy is everyone's fundamental right. It is up to a person to decide which personal information he wants to share. It is a sad truth that in India, no strict laws are enforcing the privacy of patients but shall not remain valid forever in the future. This healthcare architecture can help the patient in preserving their privacy and at the same time allowing many other healthcare services. This can help the doctors to treat the patient quickly and efficiently because they can use the medical records of that patient whose integrity is ensured by the immutability property of the blockchain. With a few improvements, this architecture can be a big help in scenarios like disaster management where user don't have to worry about the payments if he has an insurance plan registered. All the payments in such tragedy will be handled by the corresponding insurance companies. From this project, one thing that we can conclude for sure is that blockchain can provide support to our progress towards a more digitized world for business networks like health sector and therefore making a significant improvement in living standards of humanity.

There are multiple scopes in terms of improvement in the architecture. Improvement can be done in both performance and business logic. More than 1000 comprehensive performance experiments on a very similar network comprising of 4 organizations are carried out with different configurations in [Thakkar]. With common channel but two peers on each organization, they received a throughput of 140 tps. Then the article experimented with different configurations including an increasing number of channels, peers and orderers in the network, horizontal and vertical scaling of processing power and RAM on the nodes, changing block sizes, switching between the state databases, i.e., goleveldb and CouchDB and so on. After conducting all those experiments, they were able to increase the throughput to 2700 tps. The same experiments could be performed on the chaincode of our architecture to get the most optimized configuration. Further improvements can be made by incorporating banks in the architecture, which could make the payments more easier to the patients. Also, delivery agencies can also be added to the architecture to ship the medicines, lab reports, and other assets which could make the life of patients easier. Also, research needs to be done on the amount of data generated in the health sector and how can it be managed in the most efficient way.

Bibliography

[exp] Hyperledger explorer. `https://www.hyperledger.org/projects/explorer`. Hyperledger explorer main website.

[2] Hyperledger fabric. `https://www.hyperledger.org/projects/fabric`. Hyperledger fabric main website.

[3] Hyperledger fabric. `https://hyperledger-fabric.readthedocs.io/en/release-1.4/`. Hyperledger fabric documentation website.

[pte] Performance traffic engine - pte.

[Ateniese] Ateniese, G. Improved proxy re-encryption schemes with applications to secure distributed storage.

[IBM] IBM. An introduction to hyperledger.

[7] Nitin Gaur, Luc Desrosiers, V. R. (2018). *Hands-On Blockchain with Hyperledger*. Packt.

[Thakkar] Thakkar, P. Performance benchmarking and optimizing hyperledger fabric blockchain platform.