Electronic Health Record Storage using Blockchain

Amitoj Singh Chouhan
MT2017019
amitojsingh.chouhan@iiitb.org

Abhishek Rai
MT2017004
abhishek.rai@iiitb.org

May 1, 2018

Contents

1 Motivation 1

2 Objectives 1

3 Technical Explanation 2
- 3.1 Architecture . . . 2
- 3.2 Security . . . 2
- 3.3 Transactions . . . 3
- 3.4 Data Structure . . . 3

4 Technologies 5
- 4.1 Hyperledger Fabric . . . 5
- 4.2 Hyperledger Composer . . . 6
- 4.3 JavaScript . . . 6
- 4.4 node.js . . . 6
- 4.5 npm . . . 6
- 4.6 HTML . . . 6
- 4.7 jQuery . . . 6

5 Learning 7
- 5.1 Learning sources . . . 8
- 5.2 Challenges . . . 8

6 Installation 8
- 6.1 Pre-requisites . . . 8
- 6.2 Development Environment . . . 9
- 6.3 Deploying a Hyperledger Composer blockchain business network to Hyperledger Fabric . . . 10

7 Future work 10

1 Motivation

In today's world, users expect an instantaneous and seamless flow of data. Many industries have adopted or are beginning to adopt necessary technologies to guarantee their users' expectation for instant information. Unfortunately, the healthcare industry has lagged behind. Legacy systems are burdensome, slow, oftentimes vulnerable and have little role for the patient.

Health data contained in legacy systems is siloed and difficult to share with others because of varying formats and standards. In short, the current healthcare data landscape is fragmented and ill-suited to the instantaneous needs of modern users.

The relationship between healthcare professionals and patients has long been a paternalistic one. In recent times however, there has been a significant shift of authority. Medicine is being democratised and patients are more empowered.

At present, electronic health records (EHR) are stored on centralized databases in which medical data remains largely non-portable. Centralization increases the security risk footprint and requires trust in a single authority. Moreover, centralized databases cannot ensure security and data integrity, regardless of de-identification and controlled access requirements. Centralized health databases are legally a requirement and necessity in most countries worldwide and therefore require an added layer of technology to improve their portability and security.

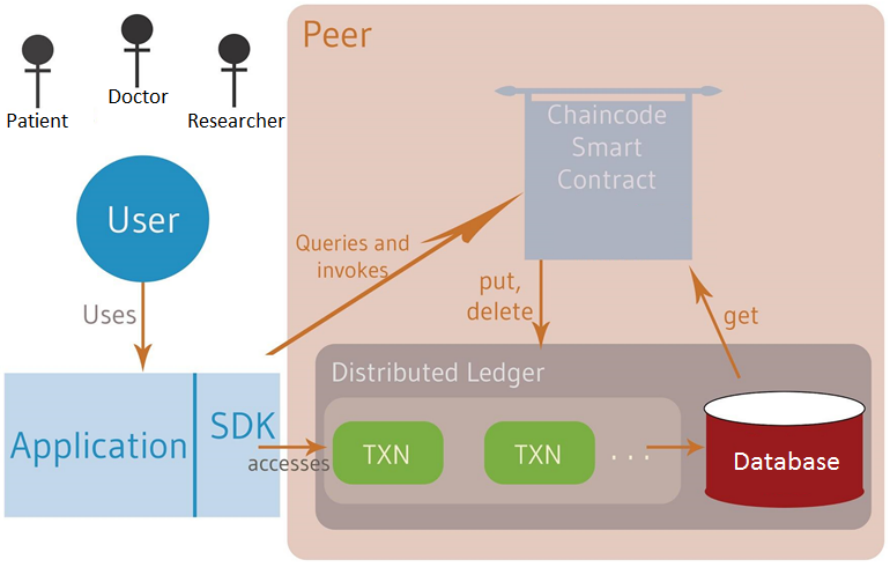
2 Objectives

- Secure, immutable and decentralized EHR database with patient owing her/his own health data
- Single version of the truth verified by the consensus of the participating hospitals
- On the tunes of upcoming digital health data bill DISHA in India
- Easy to share selective or all EHRs as consented by the patient
- Full medical history of a patient at one single point
- Easy verification of medical prescription
- Redacted EHRs for research purposes.
- Increased transparency
- No insurance fraud

3 Technical Explanation

3.1 Architecture

Application considers three types of users: Doctors, Patients and Researchers. Any one of these users interacts with our main application user interface. Next, user will invoke some query. SDK will verify global state of the blockchain and query will be submitted to the blockchain through restful service-based API. Blockchain will send the request to other peers for consensus. After the successful consensus, transaction will be submitted to the blockchain and the subsequent key-value pair will be created or modified according to the request.



3.2 Security

With creation of a new user, a 128-bit token is generated, and this token is stored in the database on the backend. Then, whenever a user logs in the application its credentials are verified, and its corresponding token is fetched from the backend via REST API. Now with every query, application passes the user's token and, on the backend REST API verifies the token and gets to know the identity of the person. This is the first layer of security.

Then when actual data is being pulled from the blockchain, blockchain verifies the user identity with the certificate it issued when the user was created on the blockchain. This forms second layer of security.

3.3 Transactions

Any interactions with health records are recorded as transactions on the network. Transactions are viewable only to the participants associated with the transaction. Here are examples of how transactions take place in the application.

Patient Granting Write Access

- Patient A grants write access to medical record to Doctor A
- Doctor A's ID is added to Patient A's authorized asset on the ledger

Patient Granting Read Access

- Patient A grants read access to medical record to Doctor A
- Doctor A's ID is added to medical record's authorized asset on the ledger

Doctor Referring Patient

- Doctor A updates the permissions to allow Doctor B to read the Patient's medical record
- Chaincode will check that Doctor A has permission to write on the medical record
- Doctor B's ID is added to medical record's authorized asset
- Patient A has to add Doctor B explicitly to his authorized list for write permission

3.4 Data Structure

Variable Type	**Variable**	**Description**
String	Record ID	A unique record ID
String	Doctor ID	Reference to a doctor
String	Patient ID	Reference to a patient
String	Encounter ID	Pointer to the encounter containing description and prescription
String	Location	Hospital where medical record is generated
String Array	Authorized	List of read permitted doctors
Date Time	Encounter Time	Time of the event

Table 1: Medical record

Variable Type	Variable	Description
String	Encounter ID	A unique encounter ID mapped to a unique record ID
String	Description	Description of the diagnosis
String	Prescription	Medical prescription

Table 2: Encounter

Variable Type	Variable	Description
String	Patient ID	A unique patient ID
String	Array Authorized	List of write permitted doctors
String	Patient Profile	Reference to the patient profile ID containing demographic data

Table 3: Patient

Variable Type	Variable	Description
String	Doctor ID	A unique doctor ID
String	Doctor Profile	Reference to the doctor profile ID containing demographic data

Table 4: Doctor

Variable Type	Variable	Description
String	Profile ID	A unique profile ID
String	Patient ID	Reference to a patient
String	First Name	First name of the patient
String	Last Name	Last name of the patient
String	Email Address	Email address of the patient
String	Address	Address of the patient
Date Time	Date of birth	To calculate age of the patient

Table 5: Patient profile

Variable Type	Variable	Description
String	Profile ID	A unique profile ID
String	Doctor ID	Reference to a doctor
String	First Name	First name of the doctor
String	Last Name	Last name of the doctor
String	Email Address	Email address of the doctor
String Array	Qualifications	List of qualifications of the doctor
String	Address	Address of the doctor
Date Time	Date of birth	To calculate age of the doctor

Table 6: Doctor profile

4 Technologies

4.1 Hyperledger Fabric

Hyperledger Fabric is a blockchain framework implementation and one of the Hyperledger projects hosted by The Linux Foundation. We are using it as the foundation for developing our application with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play. Hyperledger Fabric leverages container technology to host smart contracts called "chaincode" that comprise the application logic of the system.

4.2 Hyperledger Composer

Hyperledger Composer is a set of collaboration tools for building blockchain business networks that make it simple and fast for business owners and developers to create smart contracts and blockchain applications to solve business problems. Built with JavaScript, leveraging modern tools including node.js, npm, CLI and popular editors, Composer offers business-centric abstractions as well as sample apps with easy to test devOps processes to create robust blockchain solutions that drive alignment across business requirements with technical development. We used it to interact with our underlying blockchain.

4.3 JavaScript

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded. We used it to write our chaincode for the blockchain.

4.4 node.js

node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Node.js enabled us to use JavaScript for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

4.5 npm

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. We used it to get packages for JavaScript.

4.6 HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. We used it to create our user interface.

4.7 jQuery

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. We used it to send POST and GET ajax API calls to our back-end.

5 Learning

Blockchain: A blockchain is a continuously growing list of records, called blocks, which are linked and secured using cryptography. Each block typically contains a cryptographic hash of the previous block, a timestamp and transaction data. By design, a blockchain is inherently resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks, which requires collusion of the network majority.

Importance of blockchain: Blockchain technology has a large potential to transform business operating models in the long term. Blockchain distributed ledger technology is more a foundational technology—with the potential to create new foundations for global economic and social systems—than a disruptive technology, which typically "attack a traditional business model with a lower-cost solution and overtake incumbent firms quickly". The use of blockchains promises to bring significant efficiencies to global supply chains, financial transactions, asset ledgers and decentralized social networking.

Difference between public and private blockchain: The sole distinction between public and private blockchain is related to who is allowed to participate in the network, execute the consensus protocol and maintain the shared ledger. A public blockchain network is completely open and anyone can join and participate in the network. The network typically has an incentivizing mechanism to encourage more participants to join the network. Bitcoin is one of the largest public blockchain networks in production today. One of the drawbacks of a public blockchain is the substantial amount of computational power that is necessary to maintain a distributed ledger at a large scale. More specifically, to achieve consensus, each node in a network must solve a complex, resource-intensive cryptographic problem called a proof of work to ensure all are in sync.

Importance of Electronic Health Records: EHRs and the ability to exchange health information electronically can help you provide higher quality and safer care for patients while creating tangible enhancements for your organization. EHRs help providing accurate, up-to-date, and complete information about patients at the point of care. It helps securely sharing electronic information with patients and other clinicians. It promotes legible, complete documentation and accurate, streamlined coding and billing. It enhances privacy and security of patient data. It reduces costs through decreased paperwork, improved safety, reduced duplication of testing, and improved health.

5.1 Learning sources

- Hyperledger Fabric documentation
- Hyperledger Fabric on GitHub
- Hyperledger Composer documentation
- Hyperledger Composer on GitHub
- Blockchain online learning course on Coursera
- Medicalchain whitepaper

5.2 Challenges

- Since the blockchain technology is very new, the documentation available is limited and it is not updated regularly
- There is no proper "EHR using blockchain" sample use-case to follow
- Chaincode, Modeling, Access Control List and Query needs different syntactical knowledge

6 Installation

6.1 Pre-requisites

The following are prerequisites for installing the required development tools:

- Operating Systems: Ubuntu 16.04 (64-bit)
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x

Then you can download the required development tools using the following commands:

```
$ curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
$ chmod u+x prereqs-ubuntu.sh
$ ./prereqs-ubuntu.sh
```

Click here for more info.

6.2 Development Environment

1. Install the CLI tools: There are a few useful CLI tools for Composer developers. The most important one is composer-cli, which contains all the essential operations, so we'll install that first. Next, we'll also pick up generator-hyperledger-composer, composer-rest-server and Yeoman plus the generator-hyperledger-composer.

   (a) Essential CLI tools:

   ```
   $ npm install -g composer-cli
   ```

   (b) Utility for running a REST Server on your machine to expose your business networks as RESTful APIs:

   ```
   $ npm install -g composer-rest-server
   ```

   (c) Useful utility for generating application assets:

   ```
   $ npm install -g generator-hyperledger-composer
   ```

   (d) Yeoman is a tool for generating applications, which utilises generator-hyperledger-composer:

   ```
   $ npm install -g yo
   ```

2. Install Playground: Browser app for simple editing and testing Business Networks:

   ```
   $ npm install -g composer-playground
   ```

3. Install Hyperledger Fabric: This step gives you a local Hyperledger Fabric runtime to deploy your business networks to.

   (a) In a directory of your choice (we will assume /fabric-tools), get the .tar.gz file that contains the tools to install Hyperledger Fabric:

   ```
   $ mkdir ~/fabric-tools && cd ~/fabric-tools
   $ curl -O https://raw.githubusercontent.com/hyperledger
      /composer-tools/master/packages/fabric-dev-servers
      /fabric-dev-servers.tar.gz
   $ tar -xvf fabric-dev-servers.tar.gz
   ```

   A zip is also available if you prefer: just replace the .tar.gz file with fabric-dev-servers.zip and the tar -xvf command with a unzip command in the preceding snippet.

   (b) Use the scripts you just downloaded and extracted to download a local Hyperledger Fabric runtime:

   ```
   $ cd ~/fabric-tools
   $ ./downloadFabric.sh
   ```

Click here for more info.

6.3 Deploying a Hyperledger Composer blockchain business network to Hyperledger Fabric

Click here for full guide.

7 Future work

- EHR structure can be modeled to suit specific disease or can be modeled on established standards like openEHR
- Mechanism to access patient's medical history in emergency situations
- Insurance companies can be given limited access for fairer claims
- Pharmacists can be added to monitor medical sales