OpenEMRx: Blockchain Extenions to OpenEMR for Patient EMR Control & Privacy

A thesis submitted in fulfillment of the requirements of the degree of

Master of Technology (M.Tech)

by

Anand Kumar

Roll no. **163050089**

Supervisor:

Prof. R.K. Shyamasundar

Department of Computer Science & Engineering

Indian Institute of Technology Bombay

2018

Acknowledgements

I would like to thank my advisor, **Prof. R.K. Shyamasundar** for his valuable guidance and regular discussions and advice throughout the course of the work. I would also like to thank **Dr. Vishwas Patil**, for the help extended in the form of discussions and advice regarding the approach and methodology in realizing the aim we set forth for this project.

Abstract

In the past, the medical record was a paper repository of information that was reviewed or used for clinical and research purposes. It was severely limited in terms of accessibility, available to only one user at a time. Patients rarely viewed their medical records. Now the most of the hospitals are adopting the Electronic Health Records System(EHR), and these EHR systems have been one of the factors in transforming healthcare and health management. But still, the patients have a very small role in defining access for permission for their own Electronic Medical Records. Also, the data stored in the database is in the plain text, which can be exported by the hospital staff without patient's permission/-knowledge. Exporting and seeing patient data without permission creates the privacy problem. This motivates us to create an EHR system, which provides better privacy and control over EMR to the patients. In the dissertation, we have created a proof of concept for the Electronic Health Care Records system using blockchain with the goal of flexibility, security and preserving the privacy of patient information. In this EHR systems, we have empowered the patients by giving them the control of their medical data. The permission to access the medical data of the patient is stored on the blockchain and whenever any hospital staff wants to access the patient's data, that hospital staffs get the permissions from the blockchain and on the basis of those permissions, the staff can access the encrypted patient data.

Contents

List of Figures

Chapter 1

Introduction

An Electronic Health Record (EHR) is an electronic version of a patients medical history, that is maintained by the medical care provider over time, and may include all of the key administrative clinical data relevant to that persons care under a particular provider, including demographics, progress notes, problems, medications, past medical history, immunizations,reports.[1]

The EHR system is used to create, store and maintain the patient's medical data. There are some EHR system available, but we need a new EHR system because patients do not have visibility to the state of their personal health record neither they have access to their medical record. Patient do not have any idea of who is going to access his medical data in a hospital. Some times medical data of the patient is accessed by hospital staff, who did not even involved in providing medical care. Patients do not have any log of who got permission to see their data. They do not have any involvement in defining access policy for accessing their data. So Patients has no control over their data. Many soldiers do not take medication for post war traumas, because of privacy. Many people do not take medical help for STD because of privacy issue.

So we believe a patient should have right to decide who can see their data and who can not. Patients should also know the access policy to access their data. So this motivates us to develop a EHR system, which provides privacy to patient data but also flexibility and transparency.

We have used the open source OpenEMR EHR system. We have used the smart contract to store the access policy for access the patient data. Use of smart contract provides the

privacy to patient data, flexibility and transparency to access permission.

Chapter 2

Background and Related Work

In this chapter we shall provide the current state of data management in EMR (Electronic Medical Records) and bring out the security and privacy implications of current practices. We shall discuss several research works that try to address the security and privacy issues in EMR systems and build a case scenario for our approach.

A typical health-care system

Figure 2.1 depicts a typical HIS/EHR (Health Information System/Electronic Health Records) ecosystem. HIS/EHR play a central role facilitating health related services to the stakeholders of the system. Depending on the sophistication and design on the HIS/EHR system, the stakeholders get benefited out of it. For example, Governments rely on it to compile public health records that help them to make policy decisions for a better of its population. Pharmaceutical companies can tie up with hospitals and trial patients for testing and release of their new drug inventions and their effectiveness. General practitioners and pediatricians may act as primary precursors to potential communicable outbreaks by reporting the data to regional/national HIS/EHR. Pharmacists may provision their stocks based on the analytics reports they can run on HIS/EHR to serve better. Citizens may browse and choose health-care providers based on their past records that can be *reliably* compiled from HIS/EHR. Local ancillary service providers dependent on this sector can competitively and transparently serve the hospitals and in public welfare systems related to health. All such data-driven use cases do exist to some extent these days,

but a comprehensive, integrated system does not exist due to security and privacy issues related to the sensitive medical data present in the system. In this thesis we shall present an approach to address the security and privacy issues related to such a system. Before we present our approach, in the following, we shall highlight the potential and importance of a data-driven health-care system in which several HIS/EHR are interconnected and collaborate in a mutually beneficial way for all the stakeholders.

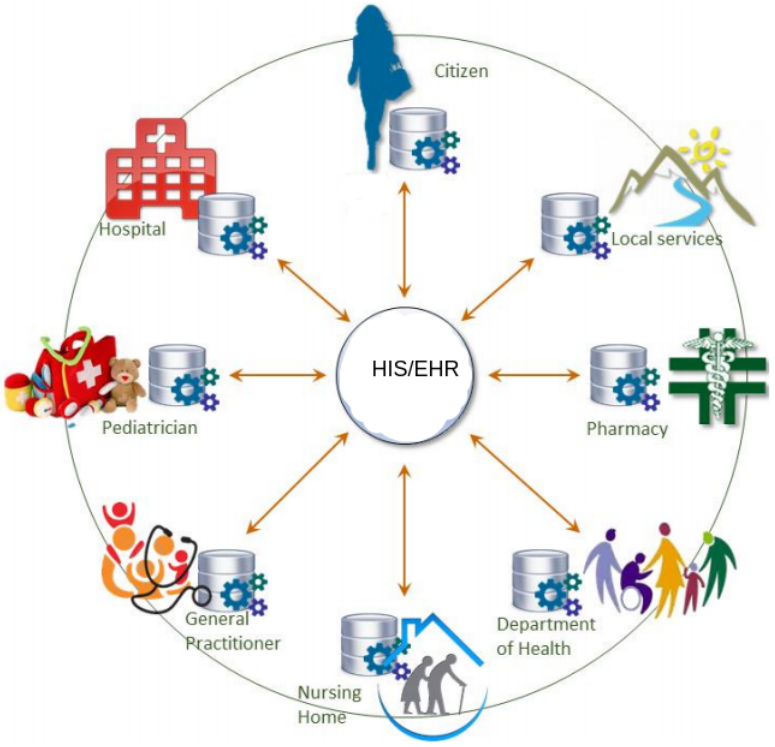

Figure 2.1: A typical health-care system

Role of *reliable data* in public health systems

Disease prevention is one of the primary goals of public health systems. In order to have a visibility of any disease outbreak or overall state of public health, the data being fed to the public health database has to be reliable. With an absence of any technological measures to prevent a malicious local database administrator, one cannot reliably guarantee the accuracy of the data being reported. In presence of a malicious health-care provider in

the system, the honest health-care providers become averse of participating in the system that cannot guarantee integrity of the data on which all participants and policy makers will have to depend upon. Therefore, providing **integrity and provenance** to the data that is being fed into the public health reporting system is very important.

Potential of *open-data* in improving health-care

Health-care systems are mainly about providing timely health services to the patients on one hand and on the other hand managing the resources required to provide those health services. Traditionally patients enter the health-care workflow system that are owned by hospitals and the agents/roles identified by the hospital – that is doctors, nurses, receptionist, pharmacist, et al interact with the patients while they progress through the workflow until they come out of the hospital. At each stage of the hospital workflow (usually called as HIS - Hospital Information System), transactions occur between the patient and the roles defined by the hospital. All such transactions and the data therein gets recorded into the HIS. HIS constitutes EMR about the patient and the meta information about the transactions, the state of resources of the hospital. All of this data in its electronic form allows the management to improve their services and provision future services. Furthermore, such HIS data across health-care providers, if *reliably* made available to national/regional health management under open-data initiative , has the following potential use cases:

1. national/regional health census (necessary for policy makers)
2. real-time visibility of health-care resources' availability (necessary for their efficient allocation)
3. performance measurement of health-care providers
4. reliable health status of patients (necessary for insurers)
5. measurement of effectiveness of drugs across large population (necessary for pharmaceutical companies)
6. transparent, competitive participation of ancillary service providers

However, when the HIS data is made available outside the administrative domain (that is a hospital), for a greater good as indicated above, a natural apprehension about data's privacy comes to the fore. Therefore, it is important to provide **privacy** to the HIS data.

Potential of *EMR portability* in improving health-care

EMR (Electronic Medical Records) of patients are the most sensitive data present in HIS. This is a PII (Personally Identifiable Information) and most of the countries have strict laws on how PII has to be gathered, stored, processed and disposed off. If a health-care information system assuages these PII requirements, EMR portability has following benefits:

1. freedom to possess and present EMR to health-care provider of patient's choice
2. well-informed decision making for medical treatment by presenting EMR for second-opinions from far away experts
3. traceability and accountability for a medical condition due to care-providers
4. reduction in time-to-treatment for emergency cases
5. improved user experience (always ready, always filled forms to get medical help)
6. informed participation in Data Economy (for monetization or for general good as listed earlier)

However, in absence of a single, common EMR data-format across the health-care providers, it is difficult to achieve *EMR portability* across the health-providers for the patients. And expecting a single, common EMR data-format is impractical due to various regulatory and operational issues that persist in real world scenarios. **The problem we have at hand is: Is it feasible to provide EMR portability for patients while ensuring complete visibility over patients' EMR w.r.t. its access control while preserving patients' privacy?**

The promise of blockchain – a new type of database

When we moved away from paper-based patient record keeping to a computer-based EHR/HIS system, all the benefits that a computer system gives over traditional paper-based record keeping system were inherited into the EHR/HIS – to list a few of those benefits: i) cost effective storage and transport of records, ii) real-time visibility of health-care workflow to its management, iii) easy and frequent report compilation over health-care data for various management and diagnostic purposes. The data is usually stored in databases that are usually maintained by *trusted* system administrators. However, visibility and access to such medical data is restricted to the people associated with that particular health-care facility. As discussed before, the potential benefits of sharing of the medical data beyond the administrative boundaries of the health-care facilities has compelling benefits for patients as well as the service providers. So far, efforts to implement such a system were marred by the challenges of data consistency, integrity, confidentiality, provenance, and control. These are well known aspects of system design that have been traditionally studied in distributed systems.

In 2009, Satoshi Nakamoto [7] presented the first working application (as a cryptocurrency called bitcoin) of a distributed database that is owned and maintained by several independent, uninitialized participants who are free to join and leave the process of maintaining the distributed database. The distributed database consists of periodic blobs of transactions, created by the participants, called as blocks and therefore the name blockchain. The blobs are public and everybody can see the list of transactions in it therefore it is also called as public ledger. Each participant checks the blob for correctness and validity of the transactions listed in it. The transactions are about transfer of value from one participant to another, who are identified by their public keys alone. The algorithm to run this type of distributed ledger show who has transacted with whom at what time is also called as blockchain. This has following inherent security properties to its transactions:

1. integrity to the transactions

2. confidentiality to the sender and recipient of a transaction

3. provable validity of a transaction
4. provenance of value with any participant
5. irrevocable nature of a transaction
6. guaranteed inclusion of a transaction

However, as described above, since the database/ledger is publicly created and maintained, anyone participating in the system can view the transactions. Therefore, the privacy of users hinges on maintaining secrecy of ownership of their respective public keys. *Bitcoin* is a greatly successful first implementation of a distributed consensus system in which a public ledger is maintained blob-by-blob fashion where creation of every blob/block is consented by the whole network. Inspired by this consensus approach several other blockchain algorithms were proposed to bring in different properties to the resulting setup – for example, permissioned vs permission-less setup, a setup having support for general purpose value transfer mechanism, condition based value transfer system, et al. In this work we shall be using a type of blockchain called Ethereum [6], which we shall introduce later in this chapter. To appreciate the properties of a blockchain based database, we shall briefly put it in context with traditional database in the following subsection.

Comparing Traditional Database and Blockchain

Database

Traditional databases use client-server network architecture. A designated authority (Admin) have the control of database. User (client) is authenticated by the designated authority, then User can modify or read the data. Since this authority is responsible for administration of the database, if the security of the authority is compromised, the data can be altered, or even deleted.

Blockchain

Blockchain databases consist of several decentralized nodes. Each node participates in administration: all nodes verify new additions to the blockchain, and are capable of en-

tering new data into the database. Blockchain database made of blocks and each block consist of some transaction(some block might be empty). For an addition to be made to the blockchain, the majority of nodes must reach consensus. This consensus mechanism guarantees the security of the network, making it difficult to tamper with. A key property of blockchain technology, which distinguishes it from traditional database technology, is public verifiability, which is enabled by integrity and transparency.

Summary of benefits of a blockchain

1. **Immutability:** In blockchain all the transactions are decentralized over all nodes, Which makes it virtually impossible to anyone to temper with it.

2. **Integrity:** Since user can see all the blockchain, so then user can be sure that he is getting unaltered data.

3. **Redundancy:** Every node in blockchain database have a copy of blockchain. So if some node looses his copy of blockchain, then it can be downloaded as the node, by re-connecting to blockchain database.

4. **Transparency:** Every user can verify, how the blockchain has been generated over time.

A programmable blockchain – Ethereum

In 2013, a group of programmers led by Vitalik Buterin proposed a flavor of blockchain called Etherem [6] that supports conditional value transfer between public keys. It provides a programming language to write such conditions called as *smart-contracts* that read past transactions on the blockchain and permit or deny current transactions that may enter into the blockchain blob. We make use of these features provided by Ethereum and design our approach to facilitate EMR portability where patients control the access to their respective EMRs with transaction privacy. We shall explain our approach in the

following chapters. Our approach is motivated by a prior work presented in [4], which we shall briefly present below.

MedRec: Using blockchain for Medical Data Access and Permission Management

MedRec offers a decentralized system for managing and transferring the electronic health record using blockchain technology, while also managing authentications,data entry, confidentiality,accountability and data sharing. This system also allow patients to access his medical information across medical care providers and patient can also transfer his previous medical record to another medical care provider. Patient leaves data across various care provider as they move on from one care provide to other care provider. This cause the patient to leave medical data across many care provider. But this system allow easy access to previous data and also encourage user to maintain their own medical data and they can review the data.

System Overview

This system uses the Ethereum blockchain and smart contract to provide decentralization and automation of some task. Medical record are stored by medical care provider and pointer to the medical record is stored on block chain with the hash of the medical record, so that user can check tampering of the medical record if happen. Patients can also share their medical data with other care providers. Patient are notified, when new data is created and its pointer uploaded to blockchain, also patient can accept and reject this new data. These are the three smart contract are used in the system.

1. *Ethereum Blockchain:* This system uses the Ethereum blockchain . The ethereum blockchain allows two type of account

   (a) *Externally owned account:* Theses account can store ether (Ethereum crypto currency), One can send ether or message and receive ether or message .One can also send transaction to the contact account.

(b) *Contract Account:* Contact live on ethereum blockchain and execute a particular code when a transaction is sent to them.

2. This system uses three contract.

   (a) *Registrar Contract (RC):* This is a global contract. This contract stored participant's identification id, their ethereum address and participant's Summary contract address. This contract should be maintain by certified institutions.

   (b) *Patient-Provider Relationship Contract (PPR):* This contract manges the medical records. This contact issued between medical care provider and patient but other can also use this contract if patient allows them. This contact stores pointer to medical data stored on medical care provider's database. Each pointer consist of query and permission on data, when this query executed on medical care providers database then it return patient data. There is cryptographic hash of data is also stored on this contract so that it can be checked that data is tampered or not. This contract also stored access information of how to access medical care provider's database

   (c) *Summary Contract (SC):* This patient's summary contract connect patient to different Patient-Provider Relationship Contract (PPR) provided by different medical care provider, So that patient can access data from all medical provide with whom patient had been interacted. Medical care provider's summary contract will have reference to patients who have shared their medical data with him and reference to the patient whom he has severed. This contract also contain status symbol which notified the patient when medical care provider adds new data or update data and this contract also allow patient to accept or reject new update. Figure 8 shows all the contract and data references.

3. *System Node:* Every medical care provider will have some Electronics Medical Record(EMR) management system. This system work well with existing EMR of medical care provider. This design include four software components both at

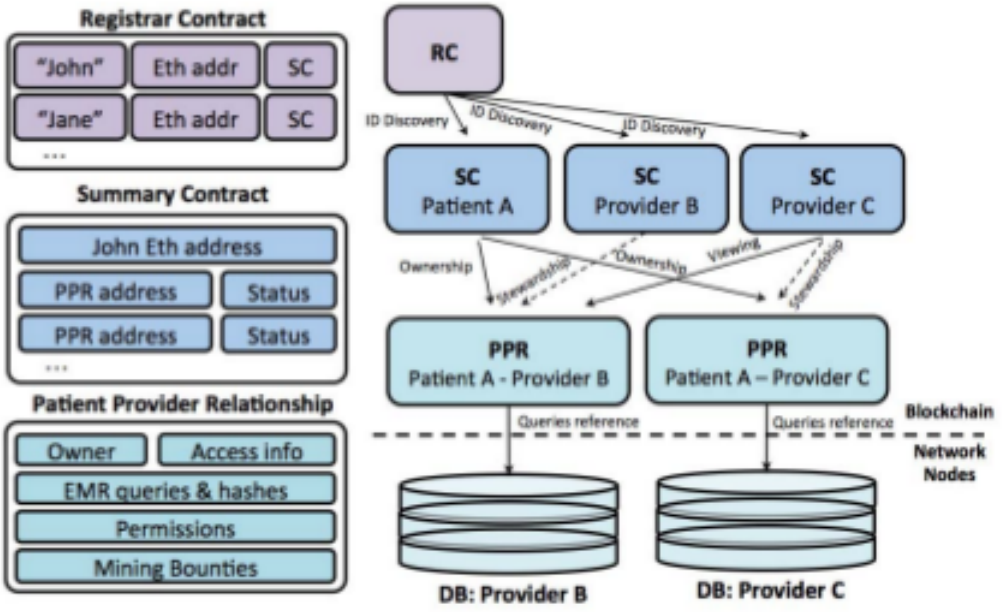

Figure 2.2: MedRec smart contracts on the blockchain and data references [4]

medical care provider and patient's node, these are ethereum client, EMR manager, Backend Library, Database Gatekeeper. Figure 9 show working of the system.

(a) *Ethereum Client:* Ethereum client is needed for interaction with ethereum blockchain and ethereum network. It also store a blockchain copy locally.

(b) *Back-end Library:* This library allows communication of EMR manger to blockchain and check whether transaction sent to blockchain network are accepted or not.

(c) *Database Gatekeeper:* Database Gatekeeper allow different parties to access database stored locally on node. It runs server to listen to queries. Database Gatekeeper checks for digital signature of parties, who want access data and only allows them if they are authorized. The request to access data, contain query and address of PPR contract. Database keeper allows to run query only if party is authorized and querying party is within permission stored on PPR Contrac.

(d) *Miners:* Miner who participate in blockchain network are get incentive for

verify the transaction and miner the block. Apart from these tradition miner, allows medical research authorities mine the network and instead incentive, medical care provider and patient release access to medical data. For this a bounty query field is in PPR contract added. when block containing transaction is mined it automatically append Block miner's address in the bounty query field as owner of bounty query of ppr contract. Then miner can query the data from provider's Database Keeper.

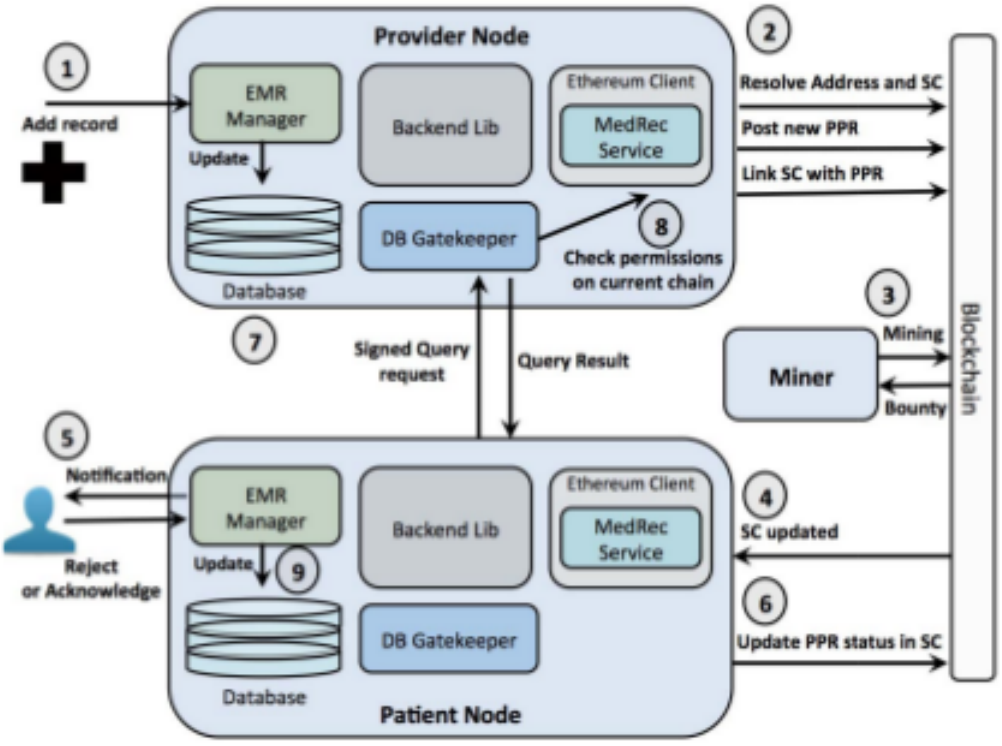

Figure 2.3: System architecture: provider adds a record for new patient [4]

How system works

Following step shows the working of MedRec system.

1. Medical care provider get the patient's ethereum address from Register contract. Create Patient-Provider Relationship Contract in this storing queries for data, per-

missions, hash of queried data, access information as to access database, patient as is owner of data. Then sent transaction to blockchain network for mining. Then provider sent transaction linking Patient-Provider Relationship Contract to Summary contract of patient.

2. When Patient-Provider Relationship Contract mined or updated summary contract get notification, patient choose to reject or accept the update.

3. Now if user want access data then, it get the query from Patient-Provider Relationship Contract and get access information for connecting the medical care provider's database keeper. It connect to medical care provider's node, and query the database, then query is verified, then patient can get the data.

4. If patient want to share the data with other medical care provider then. Patient downloads data from previous care provider. Now it can create Patient-Provider Relationship Contract for new medical care provider and link to new medical provider's summary contract. The new medical provider can download the data from patient database.

MedRec system allows a patient to share sensitive medical data between different medical care provider system also provide authentication, confidentiality, accountability, privacy using blockchain. All the access information is stored on ethereum blockchain, which provide system confidentiality, accountability, authentication.

Chapter 3

OpenEMR and Ethereum

An Electronic Health Record (EHR) is an electronic version of a patients medical history, that is maintained by the provider over time, and may include all of the key administrative clinical data relevant to that persons care under a particular provider, including demographics, progress notes, problems, medications, past medical history, immunizations,reports.[1] EHR have been one of the factors in transforming health care and health management by providing electronic access to information recorded on paper charts[8].The EHR system which is used to create, store and maintain the patient's medical data.They are tools to manage the patient data, resource and to some extent who can access what.

3.1 OpenEMR

OpenEMR is a medical practice management software which also supports Electronic Medical Records (EMR) and it features fully integrated electronic medical records, practice management for a medical practice, scheduling, and electronic billing[3]. In this project we are using the OpenEMR and implement the blockchain based access control in it. Following is the data classification in the OpenEMR and then we discussed the access policy and issue in it.

3.1.1 Data classification in OpenEMR

In OpenEMR, it is classifying the patient data into these categories: Demographic data, Medical Problem, Allergies,Medications, Prescription. The demographic data is further divided into the Who,contact, Misc,Choices, Employer,State, Guardian.

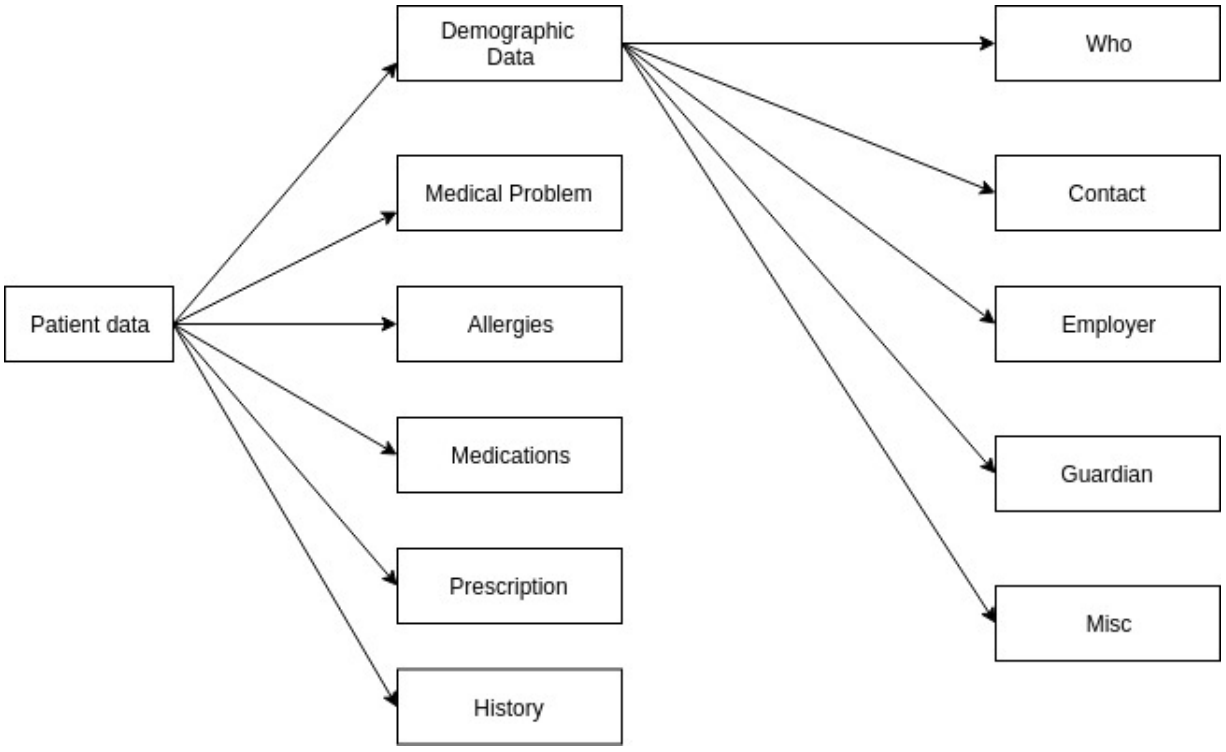

Figure 3.1: Data classification in OpenEMR

1. **Demographic** The demo graphic data is socio-economic data about the patient. It can is basic data, which can a patient provide during the registration at the hospital.
   (a) **Who** This contain the basic information for the patient identification.
   (b) **Contact** Here the patients contract detail and emergency contract detail is stored.
   (c) **Employer** This contain the patient's employer detail.
   (d) **State** This contain information about patient's race ethnicity,family size, income etc.

(e) **Guardian** This contain all detail about the patient guardian.

(f) **Misc** If patient is dead then this contain the patient diseased data and reason.

All data other than 'WHO' is optional to provided by patient

2. **Medical Problem** This contain all the data about the patient's Medical problem old and new. The previous medical problems can provided by the patient and current medical problems in the hospital are recorded here.

3. **Allergies** This contain all the allergies of the patient. It is also recorded here whether they are active or not.

4. **Medications** This contain all the medication taken by the patient.

5. **Prescription** This contain all prescription given to patient in the current hospital.

6. **History** This contain data about patient's medical history, also about family medical history and lifestyle.

3.1.2 OpenEMR Access Policy

OpenEMR has group based access policy, which is means, there are groups defined and each group has some access privileges. whenever a staff is registered to the hospital it is assigned a group. Whenever a staff person login into the system, he/she gets the privileges of the group assigned to him/her.

The more common type of the groups are:

1. Physician

2. Clinician

3. Front Office

3.1.3 Issues with OpenEMR

The main issue with OpenEMR access control is that if a hospital staff has some privileges (accessing medical data or issue prescription etc), it is for every patient in OpenEMR.

This is clearly a big privacy issue because if some hospital staff has privileged to read patient data, then he/she can read all patient's data even if they are only provide medical care to few patients. Which makes it very easy for a physicians or nurses to see the data of the patients, to whom they did not even provide the medical care. So this way, if physician has privilege to issue prescription, then he can issue prescription to any patient of any facility, but physician should only issue the prescription only to their patients. This means an ENT physicians can see or issue, prescription to a patient of orthopedic physician. So patient's medical data is not secure.

1. Any physician or some hospital staff has some privileges then it is for all the patient in the system. So if a physician has privileges to see the data then all physician can see the data of all patients.

2. Patient do not any view of which staff can access the their data and what type of data they can access.

In OpenEMR hospital staff can access the data of the patient even if the staff is not directly involved in the care providing and patient do not have any idea who is accessing his data. Patient also do not have any knowledge of access policy. These were the issue we addressed in OpenEMR and using smart contract we have solved them.

3.2 Ethereum

Ethereum is decentralized platform.[6] Ethereum has a blockchain, which is maintain by the peer to peer network. Ethereum provide a financial system and crypto currency (Ether) like bitcoin, but it also provide platform for developing the smart contract. Smart contract are program, which runs on blockchain.

3.2.1 Ether

Ether is a crypto currency just like bitcoin, this currency can be spent like bitcoin, it provide all the features of bitcoin currency. This currency is driving force of network. This is awarded ad gift to miner for doing mining.

3.2.2 Ethereum Account

There are two types of accounts in Ethereum

1. Normal or externally controlled accounts
2. Smart Contracts

Both types of accounts could hold ether balance. Transactions can be fired from both types of accounts, though contracts only fire transactions in response to other transactions that they have received. All the action happens on blockchain is due to externally controlled account.

3.2.3 Ethereum Transaction

Blockchain is made of blocks and each block contain the transactions(a block can be empty). Transactions are used to trigger the function on the smart contract or sent ether from one account to another. A transaction has all these data in it.

1. **Transaction Hash** 0x35d446711e26810e8067fe4eec462005cf44b92c840e7c8106e85c91c2994381
2. **Block Hash** 0xd709671a3d55f3da817d3fcf870071c386e4a4fd57315342e180139fa2cb0d9c
3. **Block Number** 20
4. **Transaction Index** 0
5. **From** 0x007fb6e1876788be46d3bba64ff9af61672764e0
6. **To**0xd94f7208406f25035ec41279401d98314d5cbfb4
7. **Value** 0
8. **Gas Price** 1
9. **Gas** 118392
10. **Input** 0xd897466500000000000000000000000000000000000000000000000000000000

3.2.4 Scripting Languages

Smart contract in ethereum can be written in three of languages .

1. Solidity
2. Serpant
3. Mutan

3.2.5 Smart Contracts

Smart contract are the programs that run on ethereum blockchain. After compilation and deployment on the blockchain, they reside on specific address (160 bit). Such contracts, will exist and be executable as long as the whole network exists, and will only disappear if they were programmed to self destruct Smart contract can also store ether. As the smart contract stored on blockchain then it is very hard to alter, this gives smart contract, a power to be as real contract between number of parties, can works as a trusted real party. A smart contract can receive ether and can also sent ether to other account. Condition and rule of real life contract can be coded into the smart contract,and when, those condition met, it gives some output. Every smart contract has application binary interface(ABI). The ABI is needed to access the byte code. The ABI defines which functions you can invoke as well as get a guarantee that the function will return data in the format you are expecting.

3.2.6 Interacting with smart contract

Smart contract have user defined function in that so that for interaction with smart contact, transaction are sent to smart contract from and ethereum account. For example: A permission holding smart contract for accessing the patient's data. Here then permission for accessing data are stored against the doctor nurses address on the smart contract. When a doctor sent the transaction to smart contract then he/she get the permission.

3.2.7 Smart Contract Deployment

Following steps are required to deploy a smart contract.

1. Start the Ethereum node(geth).
2. Compile and deploy the Smart contract on the Ethereum blockchain using solidity browser.
3. Get the ABI(Application Binary Interface) and contract address form solidity browser.
4. Using Ethereum web3.js JavaScript library access the function on of the smart contract.

3.2.8 Ethereum Virtual Machine

Ethereum Virtual Machine(EVM) is part of ethereum client, Ethereum client are required to connect to the ethereum blockchain. EVM is the runtime environment for smart contracts in Ethereum[2].

3.2.9 Gas

Every transaction in ethereum network will be have some fee associated with it. That fee is paid in gas, it is purchased from ether. This is for stop over-consumption of resources. Even if the transaction failed, gas is not returned to the network.

The ethereum platform provide blockchain and way to run the smart contract. These smart contract can be use to store the data, which is not easy in classical blockchain. These contract can be used to release certain information, when certain conditions are met.

Chapter 4

OpenEMRx: Using Smart Contracts to Manage EMR

As we want to protect the patient data privacy because there are several case happens, where the patient medical data is accessed by the hospital staff, who were neither involved in direct care providing to patient nor taken permission to access the data from the patient. So it is clear that the current access control is not enough to stop stop these incidents. We want to empower the patient for his medical and personal data. We want to create a system, where the patient have some control over the policy for accessing the patient data and have the access to permanent logs of permission to given to access the data to hospital staff so this was clearly beach of patient data privacy. So thats why we want patient should have control his data stored on hospital database.In our approach the access control related to patient is stored on the smart contract. So accessing patient data from the Health Care System required queering the smart contract stored on blockchain. The benefit of blockchain is that patient get the immutable logs of the all the people who have accessed his data and if patient wants then he/she can stop hospital staff to access his/her data. In this we are modify then the OpenEMR, a open source EHR software and now we are calling it OpenEMRx

4.1 Design

In our design, there is requirement of having ethereum account for everyone. Each role in OpenEMRx have has smart contract(Doctor, Nurse, Receptionist, Patient). We use the smart contract to store the permission of doctor on patient data. Every hospital staff must be registered in their respective role smart contract. We have contracts for the four roles:

1. **Physician Smart Contract** This contract registers all the Physician of the hospital. The registration is done by the Admin.

2. **Nurse Smart Contract** This contract registers all the Nurse of the hospital. The registration is done by the Admin.

3. **Receptionist Smart Contract** This contract registers all the Receptionist of the hospital. The registration is done by the Admin.

4. **Patient Smart Contract** This contract registers the patient and the medical care provider(Physician or Nurse) and the permission to access the patient data.

The Admin, Physician, Nurse, Receptionist added into their respective contract. After this if, any hospital staff, who want to access the data of a patient must be registered with patient on patient's smart contract along with the access policy must be stored on the smart contract under which they are going to access the data. Once the hospital staff registered on the blockchain, then for accessing the data, first a transaction is sent to their respective role contract if comes yes then the request is sent the patient's contract where the permission are stored and according to those permission the access to patient data is granted.

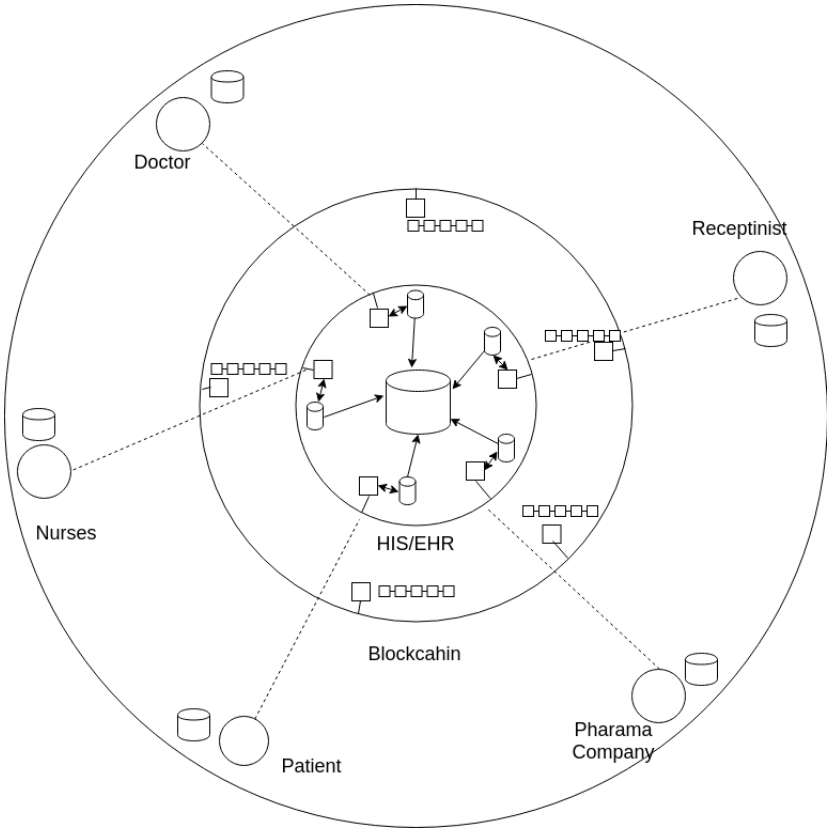

Figure 4.1: OpenEMRx

4.2 Our Default Policy

We have defined the access policy for the accessing of the patient data and it work on top of the OpenEMR's group based access control. These access policy are stored on smart contract, which is on blockchain. This theoretically it allow to have a custom policy for every patient in the hospital and gives control of some part of policy to the patient. This policy is saved on the blockchain it is always available.

4.2.1 Physician's access Policy

First of all the physician has to be registered with the patient on Patient's smart contract, to access it data otherwise it will not allow any type of access to patient data.The registration of physician with patient on patient's smart contract indicate that physician is medical care provider to patient. As the physician is the medical care provider to the patient it can access all the patient data. A physician can read, edit the Demographic,

Medical Problem, Allergies,Medications data and also can adds the prescription to the patient.

4.2.2 Nurse's access Policy

For the Nurse the to be access the data of the patient it has to be registered with patient otherwise nurse will not be allowed to access the data of the patient. Nurse is allowed to view the data demographic but not contact and guardian data. Nurse is also allowed to read the Medical Problem, Allergies,Medications data of the patient and it can also see the prescription data but can not add a prescription data.

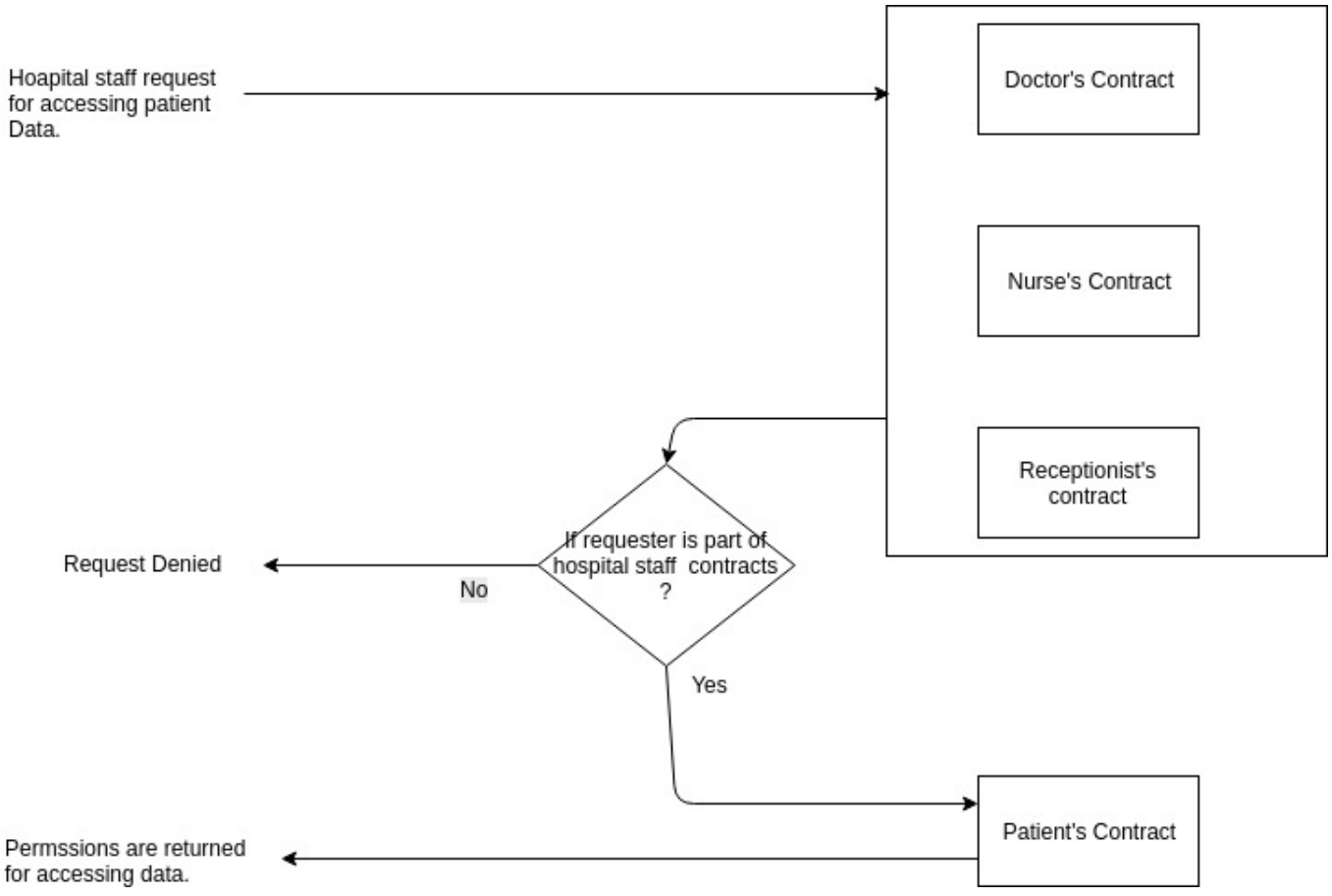

Figure 4.2: OpenEMRx Design

4.2.3 Database Encryption

All the patient data is stored on openEMR database and database administrator can see this data. Physician can also export this data. So protection from this data should be encrypted in the database. The patient's data is encrypted with a secret key and this secret is stored on the database itself. Then this secret key is encrypted with the patient's public key. Whenever any hospital staff want to access this data they have to get the permission from the blockchain, then openEMR sent the encrypted symmetric key to the patient from and receive a decrypted symmetric key form patient. With decrypted, hospital staff able to add and read old patient data. This decrypted symmetric key is not stored anywhere and nobody is allowed to see this. We are able to encrypt the medication, medical problem, allergies and prescription data.

4.3 Implementation

In this implementation first is requires a ethereum blockchain. Then the smart contract of different staff are deployed on the blockchain, then the different staff are registered on that smart contract. Whenever the patient come it is registered to patient along with medical care provider and access policy.

4.3.1 Smart Contract

Doctors' Contract

This is pseudo code smart contract of doctor. It has address of all the doctors of a hospital. This smart contract have function to add or remove doctor and it also function to check if doctor is registered on this smart contract or not. This is pseudo code of Doctors Contract

Doctors Contract

```
contract doctor
BeginContract
    declare array  doctor_array[];
```

```
    function add_doctor
    BeginFunction
    Pass In: Doctor_address, Department
        doctor_array[doctor_address]=speciality;
    Pass Out: nothing
    EndFunction


    function remove_doctor
    BeginFunction
    Pass In:doctor_address
        delete doctor_array[doctor_address];
    Pass Out: nothing
    EndFunction


    function is_doctor_exist
    Pass In:doctor_address
    BeginFunction
         if  doctor_address exits in doctor_array[doctor_address] then:
            Pass Out: true;
        else:
            Pass Out: false;
    EndFunction
EndContract
```

Nurse's Contract

Here we describe the Nurse's Contract. It has address of all the Nurses of a hospital. This smart contract have function to add or remove Nurse and it also function to check if a Nurse is registered on this smart contract or not.This is pseudo code of nurse contract

Nurse Contract

```
contract nurse
BeginContrcat
    declare Array  nurse_array[];

    function add_nurse
    BeginFunction
    Pass In: nurse_address, Department
        nurse_array[doctor_address]=speciality;
    Pass Out: nothing
    EndFunction

    function remove_nurse
    BeginFunction
    Pass In: nurse_address
        delete nurse_array[nurse_address];
    Pass Out: nothing
    EndFunction

    function is_nurse_exist
    BeginFunction
    Pass In:nurse_address
         if  nurse_address exits in nurse_array then:
            Pass Out: true;
        else:
            Pass Out: false;
```

```
    EndFunction
EndContract
```

Receptionist's Contract

Here we describe the Receptionist's Contract. It has address of all the Receptionists of a hospital. This smart contract have function to add or remove Receptionists and it also function to check if a Receptionist is registered on this smart contract or not. This is pseudo code of receptionist Contract.

Receptionist Contract

```
contract receptionist
BeginContract
    declare Array  receptionist[]

    function add_receptionist
    BeginFunction
    Pass In: receptionist_address
        nurse_array[doctor_address]=1;
    Pass Out: nothing
    EndFunction

    function remove_receptionist
    BeginFunction
    Pass In: receptionist_address
        delete receptionist_array[receptionist_address];
    Pass Out: nothing
    EndFunction
```

```
    function is_receptionist_exist
    BeginFunction
    Pass In:receptionist_address
         if  receptionist_address exits in receptionist_array then:
            Pass Out: true;
        else:
            Pass Out: false;
    EndFunction
EndContract
```

Patient's Contract

The Patients contract is for the every patient in the hospital. If patient is taking medical care from providers(Physician and Nurse). It stores the policies to access to patient for care providers. For each care provider the access policies or permissions are stored here on this contract. Any staff of the hospital access the medical data of any patient, then that staff's permission are checked on this contract. This is pseudo code of Patient contract.

Patient Contract

```
contract patient
BeginContrcat

    declare two Di-mention array patient_permission[][]

    function Give_permission_to_doctor
    BeginFunction
    Pass In: patient_address,doctor_address,permission
        patient_permission[patient_address][doctor_address]=permission;
    Pass Out: nothing
    EndFunction
```

```
function Give_permission_to_nurse
BeginFunction
Pass In: patient_address,nurse_address,permission
    patient_permission[patient_address][nurse_address]=permission;
Pass Out: nothing
EndFunction

function remove_permission_of_doctor
BeginFunction
Pass In: patient_address,doctor_address
    delete  patient_permission[patient_address][doctor_address]
Pass Out: nothing
EndFunction

function remove_permission_of_nurse
BeginFunction
Pass In: patient_address,nurse_address
    delete  patient_permission[patient_address][nurse_address]
Pass Out: nothing
EndFunction

function get_permission_of_doctor
BeginFunction
Pass In: patient_address,doctor_address
Pass Out: patient_permission[patient_address][doctor_address]
EndFunction

function get_permission_of_nurse
BeginFunction
Pass In: patient_address,nurse_address
Pass Out: patient_permission[patient_address][nurse_address]
```

```
    EndFunction
EndContract
```

4.4 Encrypted Database

We are encrypting the database of the OpenEMRx, with this encryption, if someone has access to database say database administrator, will not be able see the patient medical data. We have encrypted the patient data with a patient specific secret key and this secret key is encrypted with public key of the patient. So whenever the hospital staff requests the patient data and it passes the blockchain permission, then the OpenEMRx sent the encrypted secret key to patient and patient return the decrypted secret to OpenEMRx, with this it return the data to requesting hospital staff. When new data is added then OpenEMRx encrypts the data then added to database. With the encrypted database even database administrator can't see the patient data. Here is figure showing view of the database of database administrator.

type	subtype	title	begdate	enddate	occurrence	referredby	diagnosis	activity	pid	outcome	reaction
allergy		iodine	2015-05-13	2018-05-16	6	DR. Bafna		1	3	1	hives
medication		Norvasc	2014-05-14	NULL	4	DR. Bafna		1	3	4	unassigned
medical_problem		BCC	2014-05-06	2018-05-18	6	DR. Bafna	ICD10:C44.191	1	3	1	unassigned
medication		Norvasc	2018-06-05	2018-06-13	8	DR. Jyoti	NULL	1	1	2	unassigned
medication		sulfa	2018-06-14	2018-06-13	6	DR. Jyoti	NULL	1	1	2	nausea
medication		BCC	2017-06-06	2018-06-05	6	DR. OM	NULL	1	1	3	unassigned

Figure 4.3: Database view in Traditional OP

type	subtype	title	begdate	enddate	occurrence	referredby	diagnosis	activity	pid	outcome	reaction
medication		Metformin	2018-06-06	2018-06-14	7	www	NULL	1	1	5	unassigned
medication		mZvtVT…	gcrSPB7…	XUQzS4…	cXmDim…	biNFOy…	NULL	TnfjgiXKR…	4	9i1Lrerb3…	unassigned
medication		QD0H0…	L2Vn6Y…	bXKGiH…	HmJjsd…	u0CQy7…	NULL	AVJR6ZS…	5	HDQ+3R…	unassigned
medication		nFEui/1…	5jeCYgr4…	by1PW9…	cSe9yHJ…	/HiRvB…	NULL	HDSJR6Z…	3	AVJR6ZS…	unassigned
medication		NfQwklg…	yT0lWgN…	4tasxLiQ…	HKUyAt…	FB4RIE…	NULL	NFJSAJFA…	5	AVJR6ZS…	unassigned
allergy		8pgzti7…	5jeCYgr4…	bXKGiH…	HmJjsd…	AkPIRN…	NULL	DFNAAVJ…	4	HDQ+3R…	DyJOp8m…
allergy		WqDKfu…	FYayZ66…	jPLlolIr+…	HKUyAt…	M7pbN…	NULL	FJNDKAV…	7	3KZcoQc…	fnTtyuQN…

Figure 4.4: Database View with Encryption

4.5 Patient Portal

OpenEMR has a patient portal in which a patient can login and see their medical data. We have modified to patient portal to show, which hospital staff has given permission to access the patient data. Patient can also see, which hospital staff can read data data and which medical staff can write the data, also if patient want he/she can change the permission given to hospital staff or can completely remove the hospital staff from accessing the his/her data.

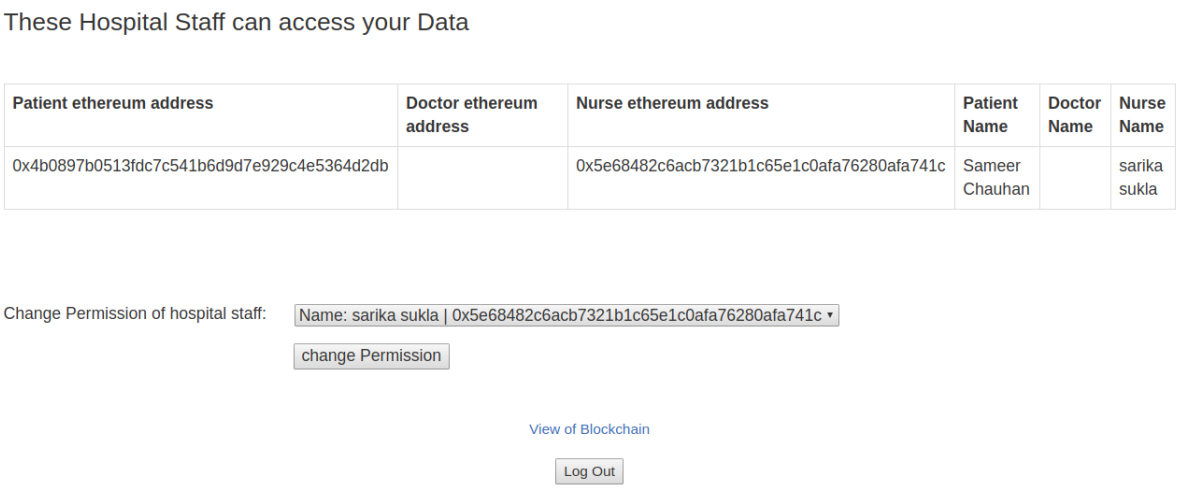

Figure 4.5: Patient Portal

Chapter 5

Different Views for the Stakeholders

As the blockchain database is available to every node in the ethereum network. This create the a single same blockchain, which is open to everyone and this blockchain can be read block by block by anyone.One can know how the blockchain is generated overtime. So to creating different views of blockchain for the different stack holders, must encrypt the data sending it to the blockchain.

We do not want to show people the ethereum address of the patient with the doctor in a transaction.So we created different views of the blockchain. So this we encrypting the patient's address with the secret key (The transaction input data) which is know only to openEMRx or the patient. So If some one goes block by block to read the transaction and able to read almost the partially decoded data, they will never know the the full decoded data.

5.0.1 External Person View

Blockchain is made of blocks and each block contain the transactions. If a person goes to blockchain and see the transaction by going block by block, he will see these transaction. This is the way he sees this a transactions. For example: This transaction is about permission given to doctor to access a patient data.

1. **Transaction Hash** 0x35d446711e26810e8067fe4eec462005cf44b92c840e7c8106e85c91c2994381

2. **Block Hash** 0xd709671a3d55f3da817d3fcf870071c386e4a4fd57315342e180139fa2cb0d9c

3. **Block Number** 20
4. **Transaction Index** 0
5. **From** 0x007fb6e1876788be46d3bba64ff9af61672764e0
6. **To**0xd94f7208406f25035ec41279401d98314d5cbfb4
7. **Value** 0
8. **Gas Price** 1
9. **Gas** 118392
10. **Input** 0xd897466500000000000000000000000000000000000000000000000000000000
    0060000000000000000000000002e2409db204425686226293fe0c0256042092ddc0000000
    0000000000000000000000000000000000000000000000000e0000000000000000000
    0000000000000000000000000000000000000000585532467364475666b583138532f3435
    677268497248614d62684d752b336c694e77654d4f754333744c314152757245645546 4f2f6
    74b41424d543859346f336178644843586c6955626a586b6d306b505a4b554c4d513d3d0000
    0000000000000000000000000000000000000000000000000000000000000000000000004
    72a64656d6f20317c7720317c6320317c636820317c6520317c6d20317c6720317c2a6c6973
    7420317c6d7020317c6d656420317c616c6c20317c6920317c2a707265736320317c0000000
    0000000000000000000000000000000000000000000

5.0.2 Nurse View

As nurse part of the OpenEMRx, it view little more the an external person. Nurse will see up to partially decoded input of transaction(as this transaction is not meant for the nurse). Nurse will get to know that it is the transaction of 'patient giving permission to a doctor'. But she can not know the public key of the patient because the patient's public key is encrypted with the secret key which is only know to patient and OpenEMRx. If this was the transaction for giving permission to him then, Nurse will be seeing the fully decoded Input.

1. **Transaction Hash** 0x35d446711e26810e8067fe4eec462005cf44b92c840e7c8106e85c91c2994381
2. **Block Hash** 0xd709671a3d55f3da817d3fcf870071c386e4a4fd57315342e180139fa2cb0d9c
3. **Block Number** 20
4. **Transaction Index** 0
5. **From** 0x007fb6e1876788be46d3bba64ff9af61672764e0
6. **To** 0xd94f7208406f25035ec41279401d98314d5cbfb4
7. **Value** 0
8. **Gas Price** 1
9. **Gas** 118392
10. **Input** 0xd897466500000000000000000000000000000000000000000000000000000000000000600000000000000000000000000000002e2409db204425686226293fe0c0256042092ddc0000000000000000000000000000000000000000000000000000000000000000e0000000000000000000000000000000000000000000000000000000000000005855324673644756 6b583138532f343567726849724861 4d62684d752b336c694e77654d4f754333744c3141527572456455464f2f674b41424d543859346f336178644843586c6955626a586b6d306b505a4b554c4d513d3d0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000472a64656d6f20317c7720317c6320317c636820317c6520317c6d20317c6720317c2a6c69737420317c6d7020317c6d656420317c616c6c20317c6920317c2a707265736320317c0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
11. **Partially Decode Input:** give_permission_to_doctor
    string _patient_address:
    U2FsdGVkX18S/45grhIrHaMbhMu+3liNweMOuC3
    tL1ARurEdUFO/gKABMT8Y4o3axdHCXliUbjXkm0kPZKULMQ==
    address _doctor_address: 0x2e2409db204425686226293fE0C0256042092DdC
    permission: $*demo1|w1|c1|ch1|e1|m1|g1|*list1|mp1|med1|all1|i1|*presc1$

5.0.3 Physician View

Suppose this physician is the physician which is providing the medical care for the patient involved in this transaction. Then the physician can see the full decoded input data. which involves the patient public key, permissions and the physician address. If this transaction does involve the physician then, it will also see only partially decoded data. This transaction include the physician.

1. **Transaction Hash** 0x35d446711e26810e8067fe4eec462005cf44b92c840e7c8106e85c91c2994381
2. **Block Hash** 0xd709671a3d55f3da817d3fcf870071c386e4a4fd57315342e180139fa2cb0d9c
3. **Block Number** 20
4. **Transaction Index** 0
5. **From** 0x007fb6e1876788be46d3bba64ff9af61672764e0
6. **To** 0xd94f7208406f25035ec41279401d98314d5cbfb4
7. **Value** 0
8. **Gas Price** 1
9. **Gas** 118392
10. **Input** 0xd897466500000000000000000000000000000000000000000000000000000000000000600000000000000000000000000000002e2409db204425686226293fe0c0256042092ddc0000000000000000000000000000000000000000000000000000000000000000e000000000000000000000000000000000000000000000000000000000000000585532467364475 66b583138532f3435677268497248614d62684d752b336c694e77654d4f754333744c31415275724564554 64f2f674b41424d543859346f336178644843586c6955626a586b6d306b505a4b554c4d513d3d0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000472a64656d6f20317c7720317c6320317c636820317c6520317c6d20317c6720317c2a6c69737420317c6d7020317c6d656420317c616c6c20317c6920317c2a707265736320317c0000000000000000000000000000000000000000000000000000000000000000

11. **Partially Decode Input:** give_permission_to_doctor
    string _patient_address:
    U2FsdGVkX18S/45grhIrHaMbhMu+3liNweMOuC3
    tL1ARurEdUFO/gKABMT8Y4o3axdHCXliUbjXkm0kPZKULMQ==
    address _doctor_address: 0x2e2409db2044256862262 93fE0C0256042092DdC
    permission: $*demo1|w1|c1|ch1|e1|m1|g1|*list1|mp1|med1|all1|i1|*presc1$

12. **Fully Decoded output** give_permission_to_doctor
    string _patient_address: 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
    address _doctor_address: 0x2e2409db2044256862262 93fE0C0256042092DdC
    permission: $*demo1|w1|c1|ch1|e1|m1|g1|*list1|mp1|med1|all1|i1|*presc1$

5.0.4 Patient View

Patient will see the full decoded input if the transaction involved this patient otherwise this will be external person view to patient.

1. **Transaction Hash** 0x35d446711e26810e8067fe4eec462005cf44b92c840e7c8106e85c91c2994381
2. **Block Hash** 0xd709671a3d55f3da817d3fcf870071c386e4a4fd57315342e180139fa2cb0d9c
3. **Block Number** 20
4. **Transaction Index** 0
5. **From** 0x007fb6e1876788be46d3bba64ff9af61672764e0
6. **To** 0xd94f7208406f25035ec41279401d98314d5cbfb4
7. **Value** 0
8. **Gas Price** 1
9. **Gas** 118392

10. **Input** 0xd897466500000000000000000000000000000000000000000000000000000000
    0060000000000000000000000002e2409db204425686226293fe0c0256042092ddc0000000
    0000000000000000000000000000000000000000000000000e0000000000000000000
    0000000000000000000000000000000000000000585532467364475666b583138532f3435
    677268497248614d62684d752b336c694e77654d4f754333744c3141527572456455464f2f6
    74b41424d543859346f336178644843586c6955626a586b6d306b505a4b554c4d513d3d0000
    0000000000000000000000000000000000000000000000000000000000000000000000004
    72a64656d6f20317c7720317c6320317c636820317c6520317c6d20317c6720317c2a6c6973
    7420317c6d7020317c6d656420317c616c6c20317c6920317c2a707265736320317c0000000
    0000000000000000000000000000000000000000000000

11. **Partially Decode Input** give_permission_to_doctor
    string _patient_address:
    U2FsdGVkX18S/45grhIrHaMbhMu+3liNweMOuC3
    tL1ARurEdUFO/gKABMT8Y4o3axdHCXliUbjXkm0kPZKULMQ==
    address _doctor_address: 0x2e2409db204425686226293fE0C0256042092DdC
    permission: $*demo1|w1|c1|ch1|e1|m1|g1| * list1|mp1|med1|all1|i1| * presc1$

12. **Fully Decoded output** give_permission_to_doctor
    string _patient_address: 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
    address _doctor_address: 0x2e2409db204425686226293fE0C0256042092DdC
    permission: $*demo1|w1|c1|ch1|e1|m1|g1| * list1|mp1|med1|all1|i1| * presc1$

If someone goes block by block to read the blockchain they will get the external person view. If someone from hospital staff try or ex-patient of the hospital want to decode the input data then they might get up to the partial decode input but never will be able to get full decoded data. So nobody from reading blockchain can find out, which patient is taking the care from which provider.

Chapter 6

Discussion

In this chapter we have put in perspective our work against the current version of OpenEMR, in terms of privacy of patient's health care data (EMR) and its access control. we have presented our findings about the computational and temporal overheads of our approach over the classical OpenEMR deployment. We have also discuss the practical impediments of our approach in real-world scenario.

6.1 Performance of OpenEMRx vs OpenEMR

OpenEMRx integrates the classical open-source version of OpenEMR with Ethereum blockchain to incorporate patient's EMR access control and transaction privacy. However, these important security and privacy feature have a computational and temporal cost on operations of OpenEMRx. We present these costs, as response time taken for a query to be executed, for a select number of operations in the below table. For direct read query the for data from database take 0.55 seconds for 1000 queries and blockchain take around 8.5 seconds for 1000 read queries. The write queries take more time 1 write query can take up to 0.2 seconds to 0.3 seconds MYSQL database and write a queries(Transaction) take upto 0.3 minutes to 2 minutes on the blockchain.

.

Operation	Response Time:OpenEMR	Response Time:OpenEMRx
New Patient Registration	1 second	1.5 seconds
Getting Patient Data	1 second	5 seconds to few minutes
Giving Permission to Doctor or Nurse to access the data	1 second	up to 1.5 minute
Registering new Staff	1 second	up to 1.5 minute

Table 6.1: Response time of operation in OpenEMR and OpenEMRx

6.2 Privacy and Control of EMR Data in OpenEMRx vs OpenEMR

1. OpenEMR have group based access control in which a group has certain privileges and each hospital staff is assigned to a group. The physician group has privilege to give prescription then the physician can give prescription to all the patients. In OpenEMRx patient has full knowledge of who can access their data and what kind of data, a hospital staff can read or write.

2. In OpenEMR, all the data of the patients is in the plain text, stored in OpenEMR database and that data can be exported by the physician. A physician can export the data of patients even without their consent. The data stored in OpenEMRx database is encrypted. So if the a physician with permission want to access data, the OpenEMRx request the key for the data from patient and after getting the key, unencrypted data is given to physician or other hospital staff. Here the patient's consent is must for their data accessing.

6.3 Impediment in real world scenario

The main problem can in real world scenario.

1. **Ethereum account** Every patient may not have the ethereum account and without that, their permission can not be stored on the blockchain.

2. **Slow Patient** A patient might provide decrypted key slowly. This will lead to long operation time.

Chapter 7

Conclusions

As our goal was to make EHR system which provide privacy for the patient data and also at provide transparency and flexibility in access policy for accessing the patient. As the only those hospital staff, who are involved in patient care, only those will able access the patient data. The change in the access policy can be made very easily, which makes our access policy flexible and at the same time the access policy is visible to patient and care providers, this makes access policy transparent. So we achieved our goal of making an EHR system which provide privacy, transparency and flexibility.

Chapter 8

Future Work

This openEMRx provides the privacy to patient data, flexibility and transparency to access policy. The transfer of key from the patient to OpenEMRx for the decrypted data can be made by an mobile application. This system can be much more benefit for patient, if it can have the MedRec capability. With the MedRec capabilities the patient can share the securely to another medical care provider into a different hospital.

Chapter 9

Appendix

.1 Demographic Table

This table stored the basic patient detail. These are the columns of that table.

id, title, language, financial, fname, lname, mname, DOB, street, postalcode, city, state, countrycode, driverslicense, ss, occupation, phonehome, phonebiz, phonecontact, phonecell, pharmacyid, status, contactrelationship, date, sex, referrer, referrerID, providerID, refproviderID, email, emaildirect, ethnoracial, race, ethnicity, religion, interpretter, migrantseasonal, familysize, monthlyincome, billingnote, homeless, financialreview, pubpid, pid, genericname1, genericval1, genericname2, genericval2, hipaamail, hipaavoice, hipaanotice, hipaamessage, hipaaallowsms, hipaaallowemail, squad, fitness, referralsource, usertext1, usertext2, usertext3, usertext4, usertext5, usertext6, usertext7, usertext8, userlist1, userlist2, userlist3, userlist4, userlist5, userlist6, userlist7, pricelevel, regdate, contrastart, completedad, adreviewed, vfc, mothersname, guardiansname, allowimmreguse, allowimminfoshare, allowhealthinfoex, allowpatientportal, deceaseddate, deceasedreason, soapimportstatus, cmsportallogin, careteam, county, industry, immregstatus, immregstateffdate, publicitycode, publcodeeffdate, protectindicator, protindieffdate, guardianrelationship, guardiansex, guardianaddress, guardiancity, guardianstate, guardianpostalcode, guardiancountry, guardianphone, guardianworkphone, guardianemail

.2 List Table

This table holds the patient's medical problem, allergies and medications. These are the columns of that table.

id date type title begdate enddate returndate occurance classification referredby extrainfo diagnosis activity comments pid groupname destination reinjury id injury part injury type celinjury gradel1 external allergyid erx source erx uploaded modifydate severity al external id

.3 Prescription Table

This table holds the prescription given by the hospital physicians. These are the columns of that table.

id, patient id, filled by id, pharmacy id, date added, date modified, provider id, encounter, start date, drug, drug id, rxnorm drugcode, form, dosage, quantity, size, unit, route, interval, substitute, refills, per refill, filled date, medication, note, active, datetime, user, site, prescriptionguid, erx source, erx uploaded, drug info erx, external id, end date, indication, prn

Bibliography

[1] Electronic health record.

[2] Ethereum virtual machine.

[3] Openemr.

[4] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30, Aug 2016.

[5] Simlindile Abongile Bantom, Retha de la Harpe, and Nkqubela Ruxwana. Accessibility to patients' own health information: A case in rural eastern cape, south africa. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '16, pages 4:1–4:9. ACM, 2016.

[6] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. 2013.

[7] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.

[8] Venet Osmani, Stefano Forti, Oscar Mayora, and Diego Conforti. Challenges and opportunities in evolving trec personal health record platform. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth '17, pages 288–291, New York, NY, USA, 2017. ACM.