Electronic Medical Record Security and Access Control Using Blockchain and IPFS

by

Md.Yeasin Ali
17101210
Suhaib Ahmed
17101099
Bany Binthe Akter
17101215
A.F.M. Mejbahul Haq
18241018

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2021

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at BRAC University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]	Bany B. Akter
Md.Yeasin Ali ID-17101210	Bany Binthe Akter ID-17101215
Suhaib Ahmed	A F M Mejbahal Haq
Suhaib Ahmed ID-17101099	A.F.M. Mejbahul Haq ID-18241018

Approval

The thesis titled "Electronic Medical Record Security and Access Control Using Blockchain and IPFS" submitted by

1. Md.Yeasin Ali (17101210)
2. Bany Binthe Akter (17101215)
3. Suhaib Ahmed (17101099)
4. A.F.M. Mejbahul Haq (18241018)

Of Fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 15, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Jannatun Noor
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

[illegible]

Dr. Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

[illegible]

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Prof. Mahbub Majumdar
Chairperson
Dept. of Computer Science & Engineering
Brac University

Dr. Mahbubul Alam Majumdar
Professor and Chairperson
Department of Computer Science and Engineering
BRAC University

Ethics Statement

The thesis is carried out in complete compliance with research ethics, policies, regulations and codes set by BRAC University. We have used various information from different sources in order to pursue the research. To collect data, we read articles, journals from different websites etc. The sources we have used here are interpreted in our own terms and are properly mentioned as reference. We appreciate and give credit to every source that helped us to continue our work. Lastly, we declare that four authors of this paper hold liability if any violation of BRAC University standard is found.

Abstract

An EMR or electronic medical records typically contains minor to sensitive medical data of a patient including their personal information, doctors' provided prescription and other physical history. Switching to this digital method from traditional reports indeed reshaped the health sector in a better way along with arising the risk of losing security. Ensuring full protection of these significant data and strengthen access control takes a fine level of structured system. To preserve the authentication, privacy and integrity of medical data in EMR systems, we came up with a model. We propose a patient centric EMR system based on blockchain technology where data are encrypted and stored in two-layered protective cloud storage. To access any data or report, physicians will be verified with digital certificates and provided permission by patient. Main purpose of this model is to serve a strong level of security to data and preserving them in a cost effective way. Keywords:

Keywords: EMR, PoA, Blockchain, IPFS, Cloud Storage, Encryption, Hash,

Dedication

We would like to dedicate our thesis to our parents who gave support by all means to come this far. Then to our respected supervisor Jannatun Noor miss and co-supervisor Muhammad Iqbal Hossain sir. We could not have conducted our research without their instructions and guidelines. Lastly, with condolence, to all the people who had and still are suffering from loss of data stealing and identity thieving issues.

Acknowledgement

In the name of almighty who gave us the strength, we are grateful to our family members. As well as to our supervisor Jannatun Noor ma'am and co-supervisor Dr.Muhammad Iqbal Hossain, the people who kept guiding us throughout the whole time of this thesis. We also appreciate our fellow team members who hold strong and united till the end to successfully complete the work. Also, our friends who helped us to remain less stressful during this pandemic. Lastly, to our university and it's authority, where we got the platform to approach one step forward to achieve our goals.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

EHR	Electronic Health Record
EMR	Electronic Medical Record
IPFS	Inter Planetary File System
OTP	Onetime Password
PKI	Public Key Infrastructure
PoA	Proof of Authority

Chapter 1

Introduction

1.1 Background

Data privacy and security is a huge concern in every possible aspect of life and when it comes to health data, it is more than a regular concern. Every other type of data might have chances to be corrected or retrieved if stolen and recovered too.Unlike those, health data or medical information are too sensitive to leak out and once they get bridged,no way to undo the damage. In this era, most of the developed country including developing ones use electronic medical record (EMR) systems which facilitated the traditional method. With technology bloom, these EHR or EMRs have became user friendly, more efficient to store and provide medical reports as well as some negative features! Amongst them most prominent drawback is unsecured data storing and lose of one's privacy threats. The reports contain sensitive medical and diagnostic details to be shared among trusted and permitted people. Needless to say, how much expensive these data could be if they are stolen and sold in black market. According to a breach barometer report, there were around 503 incidents of data breaching which affected almost 15.1 million patients in year 2018 [1]. Several security protocol have been introduced and been applied till date to build safe smooth functioned EMR systems and the process of devising is in motion. In regard to this, our system is another addition to the existing systems which is patient centric with extra layers of security.

1.2 Thesis Statement

Our model intends to be constructed with a new architectural design where EMRs will be exchanged through blockchain and IPFS cloud service with maximum security.Being it a patient-centric system, patients have the authority with whom they want to share their report or not. They will also be informed if any other user wants to get the accessibility. When health professionals request the access, patient will generate an OTP for cloud and send it to the user with the block address after system verifies the authentication certificate of the physician. To give a three layer protection, data are kept in two segments where basic medical practiced questions retain in ethereum blockchain with the encrypted address to enter cloud. As remaining every details including image files and scanned reports are stored in cloud storage which is built in IPFS cloud service.

1.3 Motivation

It is the era of rapid technological growth which is certainly a blessing for human lifestyle. However, nearly a third of analyzers think, the more digitalization occur,the more threats can come to well-being of people's lives [2]. Likewise, the increased number of usage of EMRs is no different. It took a long time for people to accept the electronic transfer of their health information due to security trust issues. At the beginning, research found consumers approve to use EMRs only when they can1 be assured of tight security and it is more proactive [3]. Thus making the EMR system more reliable has always been a sharp concern. Again, most common ways to breach medical data are the interruption of third party vendors and the medical personnel themselves [4]. Research says, around 53% incidents happen because of organizations' insiders [5]. When patients have less direct control power on their reports, confidentiality passes through more mediums which expands the risk of data stealing. This made us more motivated to focus on a system which must be patient centric. Moreover, health business started to invest in more protected EMR systems as study says loose privacy issue is one of the reason to damage the reputation of organization [6]. In addition, health records do not only contain medical valuables as those are included with one's security number, DOB, bank account number etc. For this, eminent personas and celebrities are most targeted by hackers. Many high power individuals like top employees from amazon, microsoft etc companies have faced these cyber attacks [7]. We are encouraged by these various facts that lead us to work on something affective to prevent these harmful actions and fortify user with satisfaction.

1.4 Problem Statement

We started our work with the aim to achieve the utmost positive outcome. we started to work with IPFS server which is beneficial in perspect of security as well as performance facilities. Yet many IPFS websites are protected with country lock which caused delay in our process. Moreover, EMRs are barely used in hospitals of Bangladesh thus we had difficulties in gathering medical charts.

1.5 Contribution

The concept of Medical Record System was floating on the air from a very early time which somewhat started to bring attention in 1990s when internet was emerging into the beginning of modern era [8]. Since then with the escalating development of technology, EMRs are still developing to ensure more security and protect data hacking. However, none of them could claim to stand out the issues and guarantee the safety protocols. Even some of those discard the standard at the interface designing level which leads to loose security concern [9]. With the upgrades to the existing models, it came this far of using blockchain based patient-centric EMR systems. We studied a system like that [10] and developed a better architectural system with an addition of storing data in IPFS cloud. IPFS is a distributed network which creates numerous number of blocks alongside generating unique hash values when a data is uploaded.This ensures that even if info are being hacked, attackers might get only a part of that.

Chapter 2

Background

2.1 Background

The primary requirement for this paperwork is to secure the private health data of the patient and make it patient centric while providing accountability and non-repudiation for the physicians. The main reason behind choosing this was the threat of a hack. On the black market, electronic medical health records (EHR) could be worth hundreds or even thousands of dollars depending on the contents [23]. There were multiple occasions in the past where this has come to pass. The number of annual leaks of health records increased 70 percent to 344 over the past seven years, according to a recent report in the Journal of the American Medical Association, With 75% of records being either breached, destroyed, or stolen and 132 million from a "hacking or IT incident" [24]. Also more than 2,000 breached documents containing 176.4 million records were reviewed by researchers at the Massachusetts General Hospital Center for Quantitative Health as they were reported to the Department of Health and Human Services between 2010 and 2017 [24]. This begs the question, why is health data such a favorite target of the hackers? Because it provides enough data to fully steal the identity of a person and commit a wide range of other crimes [25].

Access control based on Blockchain, introduces different modern e-health security features with great advantages over traditional access control solutions. Firstly, the blockchain creates permanent transaction ledgers for data sharing applications [21]. This guarantees that transactions recorded in the blockchain will not be modified or manipulated by some party and transactions are only written to the blockchain when recovery operations are not allowed. Secondly, transparency and the ability to solve the problem of data leakage, can be achieved using blockchain access control. Any unauthorized access to data storage would be mirrored on the blockchain and broadcasted to all network members. In this way, any blockchain user can track data access and identify malicious transactions. Thirdly, the authentication and user identification features can be achieved by using blockchain-based smart contracts [22]. Smart contracts will authorize effective user access to health data storage by implementing stringent access control policies, as well as identify and avoid possible threats to health networks in a distributed manner. Finally, Blockchain decreases reliance on central repositories in compliance with smart contract technology to guarantee fairness between parties in transactions. Since the smart contracts on the blockchain are public, all the linked entities on the blockchain network would have a copy of them, which would allow all contract activities an equal right.

So, we took it upon ourselves to put electronic medical records in a secure place and incorporate an accountability system to ensure only authorized medical personnel can access it. Recent EMR management methods consist of merely storing medical information in a blockchain [26] and establishing an omnipresent social network that enables users to exchange data gathered and preserved in a blockchain through medical sensors [27]. A distributed electronic medical record system using blockchain technology and an attribute-based signature system with multiple authorities were proposed in another proposal where a patient endorses a transaction according to the given attribute while disclosing no information other than the evidence that it has been attested to. While the scheme gives a patient the opportunity to manage and exchange their medical records safely, it imposes an overhead requiring several authorities to sign the transaction. In comparison, the confidentiality of the data contained in the blockchain [28] is not addressed. Then, a new solution called BlockIPFS was introduced to reduce the fundamental security and access management issues and establish a simple audit trail, but it only managed to push metadata of the files added into the distributed file system[20].

We are designing an electronic medical record solution in our proposal whose access control is patient-centric. Our strategy focused on both blockchain technology and IPFS cloud infrastructure. We suggest a stable distributed EMR that needs a basic infrastructure, and offers uninterrupted access control, with little overhead, unlike the previous works.

2.2 Blockchain

Blockchain, though that's not just it's only implementation, is considered the spine of bitcoin. It's a ledger that is encrypted. A block in the blockchain is a list of all the latest and confirmed transactions that have occurred recently. And it becomes a permanent block in the chain once confirmed.

It is built with 3 technologies -

1. **Cryptography -** Apart from the very advanced hashing algorithm, Blockchain uses a mixture of public key and private key. One key to encrypt another to decrypt. Asymmetric encryption.

2. **P2P Network -** Ensures complete consistency. No change possible as verification gets rejected.

3. **Blockchain Program -** It has a lot of protocols and security features based on the requirements. Can be implemented in any language.

The architecture of a broadly specified blockchain framework is shown in Figure 2.1. All P2P network members here, while synchronizing with other peers based on a consensus paradigm, need to store blockchain data on their own framework. The consensus is, in fact, characterized by most peer nodes' longest chain agreed upon.

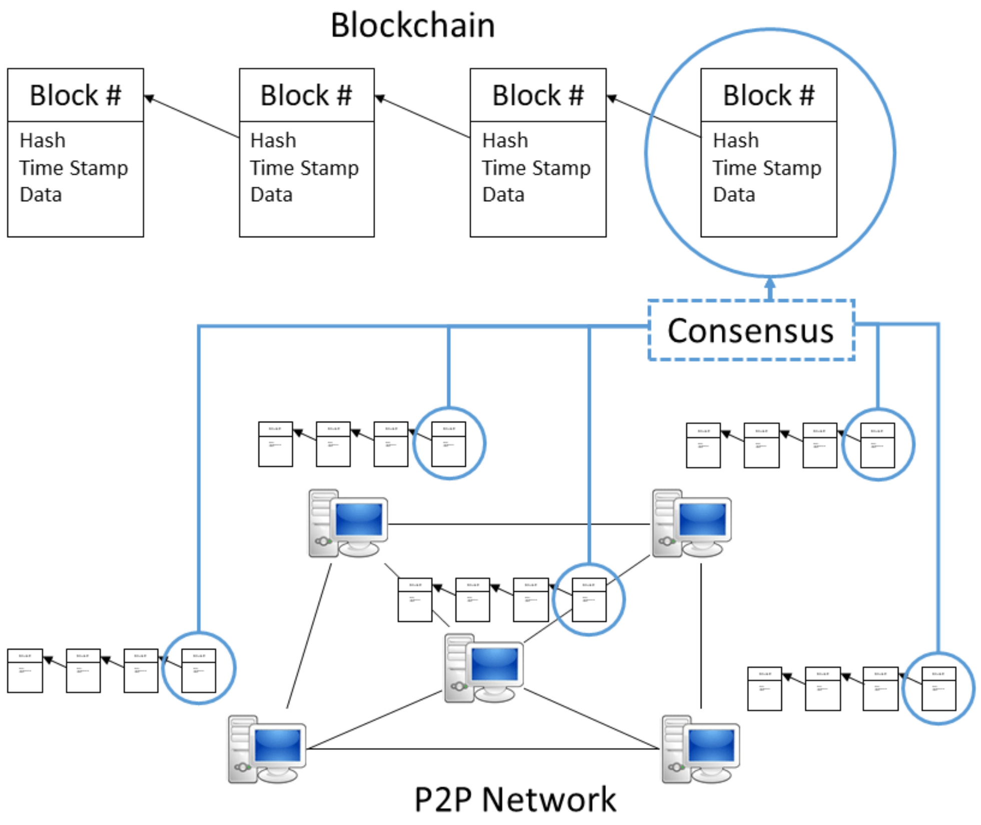


Figure 2.1: Key Element of A Blockchain System [11]

The representative of a classic blockchain system is Bitcoin [12]. It has drawn more than ten thousand nodes as the first decentralized ledger to create the highest market capitalization among all cryptocurrencies. Bitcoin's most notable achievement is that it addresses the double expenditure challenge in order to make digital currencies exclusive and meaningful. The popularity of Bitcoin, in fact, opened the door to the public for blockchain applications.

Ethereum[13] is designed as a platform for facilitating decentralized smart contracts with its own currency vehicle, Ether, in order to add more value to the blockchain ecosystem. The notion that it is possible to automatically notarize and fulfill legal contracts, is referred by Smart contract [14] refers to t. A set of smart contracts written in blocks can be implemented by Ethereum developers when equipped with Solidity[15]. Because of its everlasting nature, Ethereum extends the application of blockchain from the data to the computational realm. In other words, once the developers have compiled and deployed their programs to the public, no one will ever revise the program's logic. Publishing a smart contract therefore provides public users with a set of trustworthy functions. The smart contracts would be implemented in a decentralized fashion by the distributed nodes when invoked.

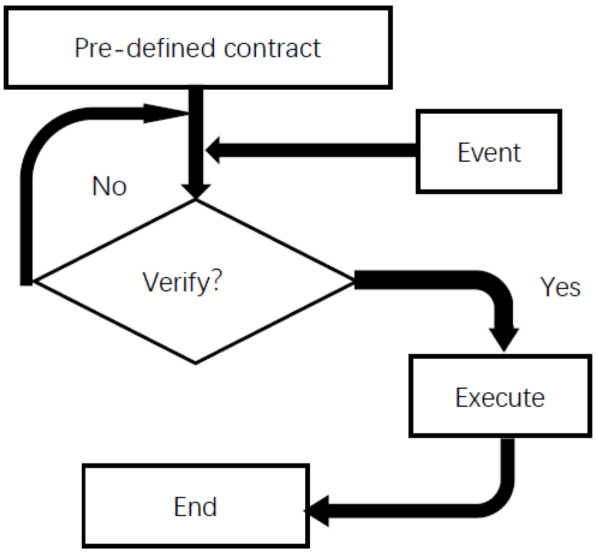


Figure 2.2: Flow chart of the smart contract [16]

There are three different types of blockchain currently available -

1. **Private Blockchain -** A private blockchain has a system of access control which runs within a single entity. Participants need to be accepted, and future entrants will be determined by current participants.

2. **Consortium Blockchains -** Consortium blockchains are, from the viewpoint of some individuals, a subclass of private blockchains. "They are also often referred to as "partially private". Likewise, it has many of the same advantages as private blockchains, such as high performance, high scalability, and better privacy of transactions than permissionless blockchains. However, rather than allowing an entity in direct charge of the blocks, the blockchain consortium is a system where a pre-selected group of nodes manages the consensus method.

3. **Hybrid Blockchains -** This type of blockchains are the mixture of a blockchain with permission that is controlled by one or more parties and a blockchain without permission that is not controlled by either entity, but approved by the majority of network users. Through an immutable record on the permissionless blocks, it can render the transactions private while also verifiable.

Many blockchain implementations with slight modifications in key features are always being released to fix vulnerabilities in the currently available structures. Someone may want a robust implementation but also can be easily changeable. It can be determined by checking the stats of the networks git repository. It might also be required that the repository has an active developers community - it can be easily determined by checking the contributor, code commits and branches stats. When choosing a suitable technology for implementation, any project might be subject to a similar comparison and analysis.

2.3 IPFS Cloud

IPFS is a distributed system where files, websites, programs and data can be stored and accessed.. And according to Zheng [17], IPFS is the best solution to cope with the ever growing storage needs. In 2018, the ledger size of bitcoin blockchain had exceeded 200GB and none of this data could be deleted due to the features of blockchain technology. So, they showed that if the data were directly deposited into the IPFS network by the miners and the returned IPFS hash of the transaction was stored into the block then the new data size could be greatly reduced due to the characteristics of the IPFS cloud.

Three fundamental principles to understanding IPFS are [18] -

1. **Unique identification via content addressing -** IPFS follows unique preferences and conventions for data-structure. IPFS uses these conventions in order to generate an IPFS address from the dumped content that uniquely identifies it on the IPFS network.

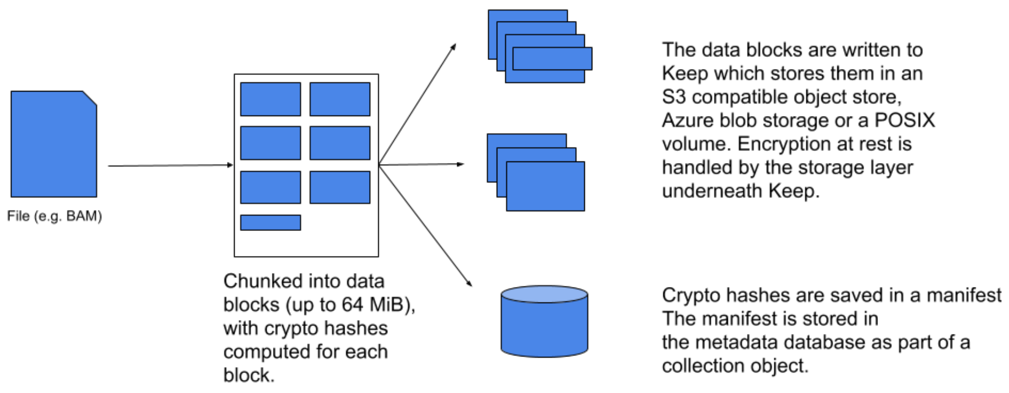

File (e.g. BAM)
Chunked into data blocks (up to 64 MiB), with crypto hashes computed for each block.
The data blocks are written to Keep which stores them in an S3 compatible object store, Azure blob storage or a POSIX volume. Encryption at rest is handled by the storage layer underneath Keep.
Crypto hashes are saved in a manifest The manifest is stored in the metadata database as part of a collection object.

Figure 2.3: Content Addressing

2. **Content linking via directed acyclic graphs (DAGs) -** IPFS takes advantage of directed acyclic graphs (DAGs). It mostly uses the Merkle DAGs, in which each node has a unique identifier that is a hash of the content of the node. In a Merkle DAG, IPFS gives CIDs to a content and ties the content together.

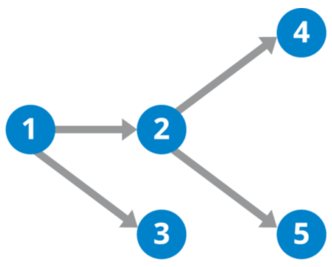
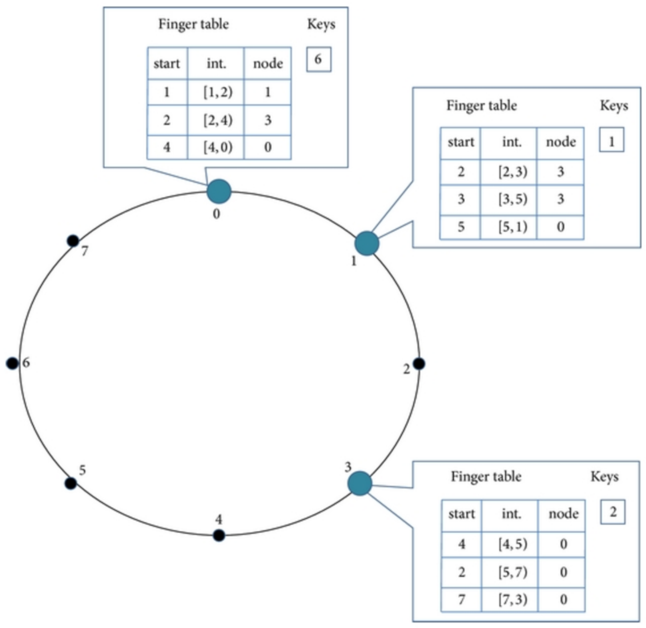


Figure 2.4: Directed Acyclic Graphs

3. **Content discovery via distributed hash tables (DHTs) -** IPFS uses a distributed hash table or DHT, to find the peers that host the content. A database of keys is a hash table. The table where it is divided across all the peers in a distributed network is a distributed hash table.



Finger table | Keys: 6

start	int.	node
1	[1,2)	1
2	[2,4)	3
4	[4,0)	0

Finger table | Keys: 1

start	int.	node
2	[2,3)	3
3	[3,5)	3
5	[5,1)	0

Finger table | Keys: 2

start	int.	node
4	[4,5)	0
2	[5,7)	0
7	[7,3)	0

Figure 2.5: Distributed Hash Tables

Chapter 3

Proposed Model

3.1 System Model

In our proposed model of electronic health record system, we designed the whole system using modern technologies like blockchain, cloud and cryptography. The system itself is complex and hybrid. It supports an advanced infrastructure system which is called "Public Key Infrastructure" (PKI). This assures the security measures of such a system that works on the public-key mechanism as trustworthy and preserve the confidentiality of private key and the integrity of the public key [29].

In our model, we used this public key infrastructure for the authentication, verification and building trust among the parties that want to transfer data among each other. In addition to that, we used a session key which is the most important element of our proposed model. This key is similar to a symmetric key and controls the access to data that is stored in both blockchain and cloud storage. Most importantly,this makes the system patient-centric. The only patient has the ability to share this key with others. Thus it preserves the ownership of data of patients. Moreover, each physician who wants to view or store any data of patients must have the consent of the owner and they need to prove their identity to patients so that patients can build trust upon them. This is done through the use of digital certificates which are issued by the official certification authority (CA). Not only physicians but also patients are also issued this certificate as they also need to identify themselves as genuine and trust-worthy for the physicians. Added that, we used hash checking and OTP mechanism to make our system more reliable and safe for data.

In public blockchain, the consensus is highly computationally-demanded. As a result,probabilistic consensus such as Proof-of-work is costly for the miners as all the miners compute each other to solve the complex mathematical puzzle which results waste of resources and needs high processing power. Moreover, consumes a huge amount of electricity [29]. For that, we used Proof-of-Authority (PoA) which is a reputation based consensus and not computationally demanded like probabilistic consensus. Furthermore, In PoA, blocks and transactions are validated by accounts that are approved before.

Figure 3.1 illustrates our proposed model of an electronic health record system. The overall process is done in three blockchain transactions. Here we can divide the user of our systems into two actor categories. One is a patient and another one is the physician. All the healthcare institute, doctors, healthcare service provider is categorized as physicians.

At first, all need to register in the system. In the first place, when a patient vis-its a physician, he/she generate a session key which is the core element for gettin gaccess to any data of the system. After generating the key, the patient encrypts this key with his public key and some data and stores it in the blockchain. Here those data includes cloud links where physicians will store his/her medical record. This also includes the previous hash value of the stored cloud data. However, in the first place,the patient has to store the hash value of the empty file since initially there exists no record in cloud storage. This is transaction number one where the session key is stored safely inside the block.

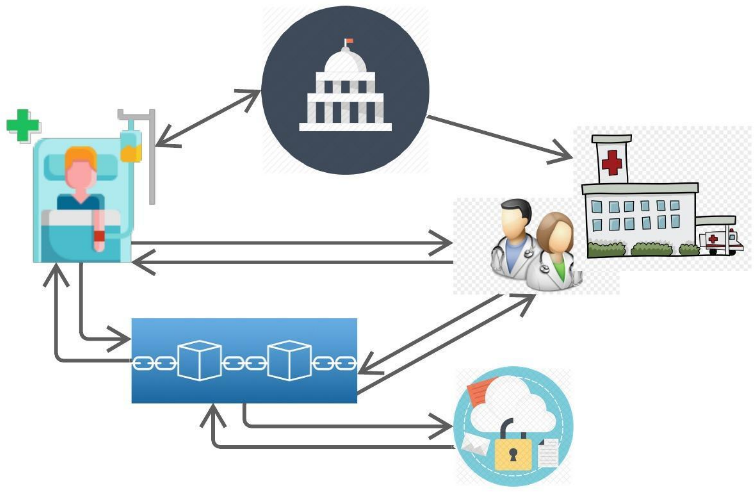


Figure 3.1: System Model

Onwards, if any physician wants to access or store data of any particular patient, first he sends an access request to that particular patient for allowing him/her to grant access to the medical record. The physician sends his/he rdigital certificate attached to the public key to the patient. It is then checked by the local machine of the patient. If the certificate is verified, this means that the request comes from the right person in right place. This makes patients build trust with the physicians and so, the patient becomes ready to build a secure channel with the physician to send necessary data. For that, the patient retrieves his previously encrypted key from the transaction simply by addressing his public key. Then he decrypts the data and key and thereafter encrypted them again. However this time he uses the public key of the physician instead of his own. After encryption, he again stores the data into the blockchain. This ends the second transaction of our proposed system.

While the second transaction is being operated, the patient generates a one-time pass-word (OTP) and encrypts it with the recipient's public key. Afterward, sends it to the recipient. Because of the secure channel built between patient and recipient at the beginning, we can trust the data transportation. However in this case even if someone breaks into the secured communication channel and stole the OTP, this will not be of any use as the data are encrypted with the recipient's public key and can only be decrypted with his public key. Moreover, this OTP will be used for access to the cloud.Fortunately, even by using only OTP, one cannot enter cloud storage as it needs some more verification of data that exists in the blockchain

After all of this, now physicians have the OTP, blockchain data decrypted by his private key. Here blockchain data consists of some medical records, cloud data address and the previously stored hash of the cloud data. Finally, he/she has access permission to view both blockchain and cloud records

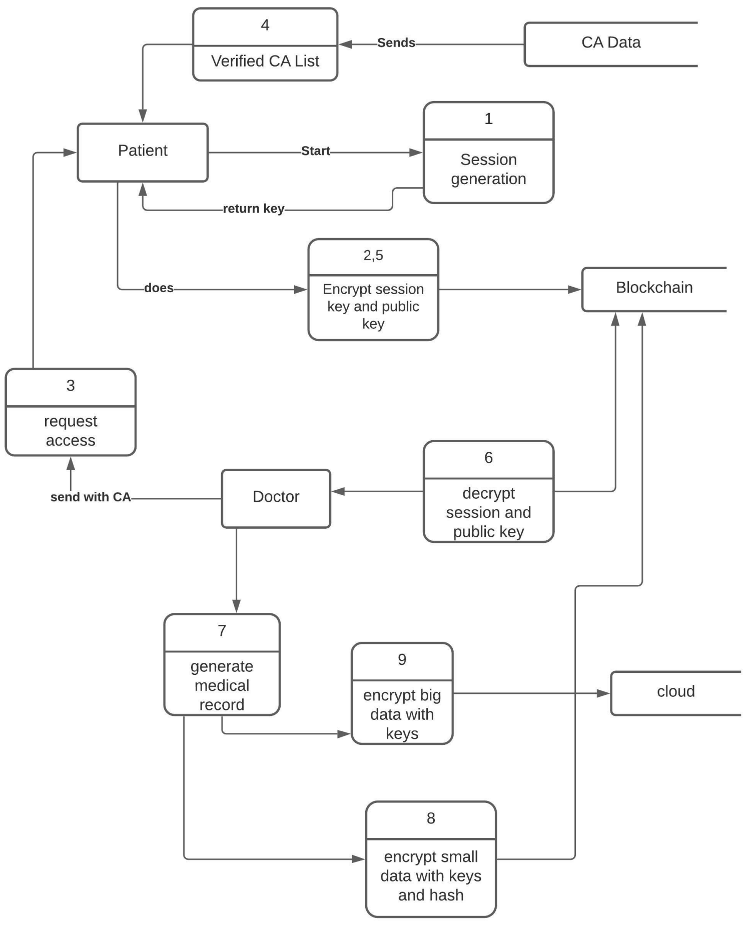


Figure 3.2: Flow of data the mode

If a physician, needs to store newly generated medical records then our third and the last transaction comes into role play. We categorized the generated data into two categories. One for storing into blockchain and another one to store into cloud storage. For the data that needs to be stored inside cloud storage, physicians encrypt that data with the public key and session key of the patient. This returns a hash value. Following this, the remaining data, cloud data location address, hash value and session key are encrypted with the patient's public and physicians sign the encrypted data with his private key. Finally, the data is sent into the blockchain to create anew transaction and this ends the third and last type of transaction of our proposed system. Figure 3.2 has a clear view of how the proposed model sends and retrieve data among various process, storage and actors.

3.2 Sequential Activity of the Model

To build a clear overall understanding of the model, it is important to understand the activity of the model sequentially. This helps to create a good visualization of the system model.

Figure 3.3 show each activity of the system that needs to be done and check by all types of actors. This clearly shows that every actor must need to be registered to use the system. Users then can generate a session key or can receive request access from the physician. If he wants to accept the request, the physician must be verified by the architecture. After that, he will make the session key to be ready for use by the physicians. For this, he will send OPT and blockchain addresses to the physician

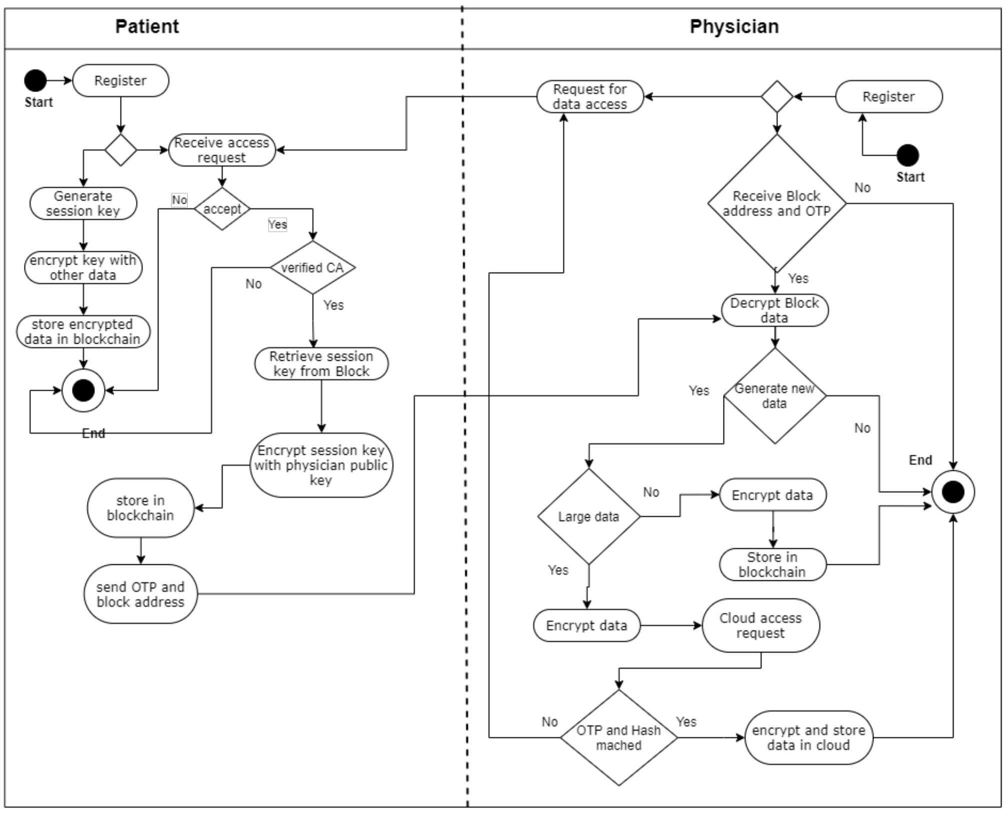


Figure 3.3: Activity Model of the System

where he will get the session key that was mentioned before. On the other hand,physician, either request access for the medical records or if already allowed can visit the cloud and blockchain by using the OTP and hash. If they do not get matched the system will simply pull him from the server. If get matched, data can be stored by encrypting data and key together.

3.3 Stored Data of IPFS

To ensure data security and immutable data store capability we designed an Ethereum blockchain-based system to store the medical data of the patient to the blockchain. We used text- based data here. However Image-based data can be very costly and-can make the overall project more costly. The small data which will be stored in the blockchain are

- History of present illness.
- Physical Examination: This includes vital signs, muscle power, organ system examinations, etc.
- Assessment and plan: These includes recent diagnosis and treatments
- Physicians order and prescription.
- Progress notes.
- Test Results.

The other types of data like medical chart [31] which will contain records of every health- relevant event that has happened to a patient since birth which includes multiple types of files such as patient's prescription as a document, images of the test report such as MRI or X-ray, as well as some other formats of data, will be stored in the Inter Planetary File System (IPFS) [32]. Every document or file is going to be identified by a unique hash value by which the doctor and the patient can access the record. As IPFS works as a decentralized storage system this record will be divided into small pieces and will be distributed to multiple peers as a chunk of the whole data around the world. So it will be almost impossible to delete or change the data as a hole

A medical chart in cloud storage will categorize in such a way that the doctor or the patient can easily find what they need to know. We propose is to categorize the storage in the following way:

- Surgical History: This directory will include operation dates, operation reports,narratives, etc.
- Obstetric History: This directory will include documents of pregnancies, any kind of complications, outcomes, etc.
- Medical Encounters: This directory will include documents like hospital admissions, specialist consultations, and routine checkups.
- Immunization Records: This includes all types of vaccinations, immunoglobulin tests, etc.

- Medications and medical allergies.
- Family History: This directory will include the patient's immediate family member's health status, cause of death, common family diseases, etc.
- Social History: This includes close relationships, past and current occupations,etc.
- Habits: This includes normal habits such as smoking, alcohol consumption,exercise habits, diet, sexual history, etc.
- Development History: This directory includes documents of the patient's growth chart, motor development, cognitive/intellectual development, social-emotional development, etc.
- Demographics: This includes the patient's race, age religion, occupation, another contact information.

3.4 Patient-centric access control

Our main focus for this approach is to make electric medical data more secure and patient- centric. To do that we will use three-layers of access points to secure the data and easy for the patient to know whom to share, how and who is viewing his data.On the first layer, the physician will be given a session key by the patient himself.We will use the certificate of authenticity (CA) to make sure that communication is going on between the patient and the physician. This session key controls access to both blockchain and cloud data. On the second layer, the physician will be asked for the previous hash value of the stored cloud data. The physician will get that hash value from the blockchain by decrypting the data and will put the hash value into a dialog box. Our model will then cross-match that hash value with the last uploaded hash value from the storage. Along with that, the third layer works. In this layer,the patient generates a one-time password (OTP) when the physician request access permission for the data. This will be sent to the physician by a secure communication channel build by the PKI infrastructure. Afterward, the physician will use this one-time password to get access to the cloud. It is important to mention that both hash and OTP are needed to access the stored data of the cloud storage. Thus, this makes our system patient-centric as OTP, cloud data hash and the session key is controlled and can only be shared by the patient himself and this ensures the data has not been tempered.

3.5 Data Security

Some of the text-based files will be stored in the blockchain. These data will contain some important information about the patients. To secure that we will use public-key encryption. By using this technique of cryptography, the patient can control who is viewing the data. In addition, it is non-temperable inside blockchain by the architecture of blockchain. Since all the large files will be stored in IPFS, anyone can access the files if anyhow they can manage or generate and get the hash value which can defeat our purpose of securing the patient data. To secure this problem, the files which are stored will be encrypted using asymmetric encryption [30]. This encryption technique uses a pair of private and public key algorithms for securing the data. This way only the authorized user can access the files and view them.

Chapter 4

Experimentation

4.1 Configuration

For the experiment of our proposed model, we used the ethereum platform and IPFS for cloud storage. The local computer which was used for this experiment has the following specification :

Hardware-

- Processor: Intel Core i5-8500
- CPU: 3.00 GHz
- RAM: 16.0 GB
- System Type: 64-bit, x64-based processor
- GPU: Nvidia GeForce GTX760

Operating System-

- Windows 10 Pro
- Version 2004
- OS Build 19041.685

In addition, for the experiment we used the following software stacks :

- Remix IDE: Online browser-based IDE for building and testing the smart contract
- IPFS: Inter Planetary File system
- IPFS Desktop
- NodeJS: Javascript runtime environment used for running geth js console
- Geth: Command-line interface for ethereum
- Puppeth: CLI wizard for the ethereum network

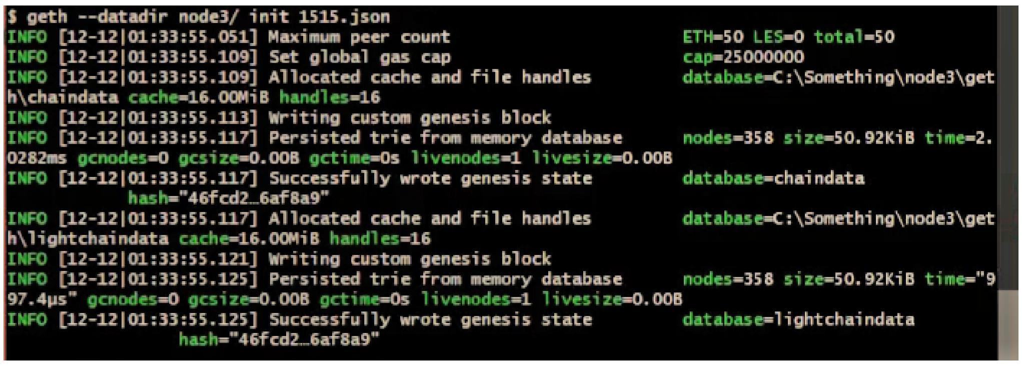
- Ethereum Network Stats: Visual interface for tracking ethereum network status
- Git: Provides an emulation layer for a Git command-line experience
- Sublime Text 3 editor

4.2 Experimental Approach

We have done this experiment in two-part. First, we have implemented the blockchain part with ethereum and geth. Then we have implemented the IPFS part using node.js and IPFS desktop version.

4.2.1 Blockchain

We created a private test network so that the test process can be monitored from the perspective of multiple blockchain nodes. For ease of activity, we used “ethereum” which is a global, decentralize, open-source cryptocurrency and blockchain platform. To experiment with our model we used geth also known as “Go Ethereum”. It is a command-line interface for running the ehtereum node that is implemented in the “Go” language. The version of geth that we installed was 1.9.25-stable-e7872729 and the Javascript console of geth ran by nodeJs version-14.15.3. We created a genesis file for our blockchain using puppet and it was built in under Proof-of-Authority(PoA) consensus engine. Initially, we created three blockchain nodes each consists of one account. This account has a balance of 20^{43}th which is more than enough for testing purposes. Afterward, each of these nodes was successfully written in the previously created genesis file.



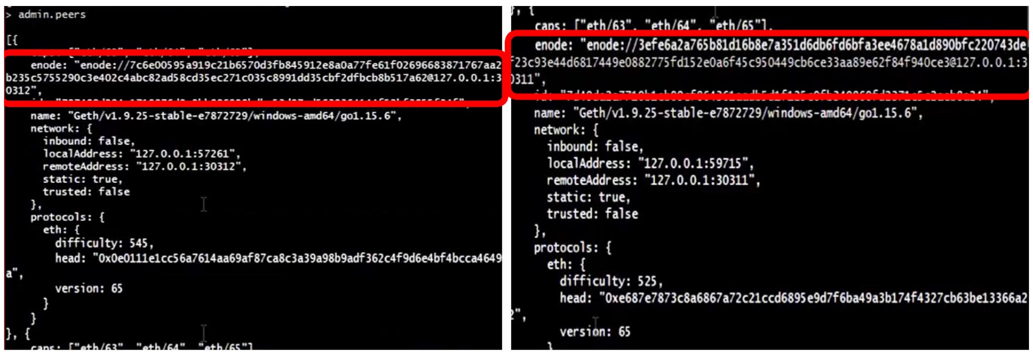
```
$ geth --datadir node3/ init 1515.json
INFO [12-12|01:33:55.051] Maximum peer count                       ETH=50 LES=0 total=50
INFO [12-12|01:33:55.109] Set global gas cap                       cap=25000000
INFO [12-12|01:33:55.109] Allocated cache and file handles         database=C:\Something\node3\get
h\chaindata cache=16.00MiB handles=16
INFO [12-12|01:33:55.113] Writing custom genesis block
INFO [12-12|01:33:55.117] Persisted trie from memory database      nodes=358 size=50.92KiB time=2.
0282ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-12|01:33:55.117] Successfully wrote genesis state         database=chaindata
          hash="46fcd2…6af8a9"
INFO [12-12|01:33:55.117] Allocated cache and file handles         database=C:\Something\node3\get
h\lightchaindata cache=16.00MiB handles=16
INFO [12-12|01:33:55.121] Writing custom genesis block
INFO [12-12|01:33:55.125] Persisted trie from memory database      nodes=358 size=50.92KiB time="9
97.4µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-12|01:33:55.125] Successfully wrote genesis state         database=lightchaindata
               hash="46fcd2…6af8a9"
```

Figure 4.1: Genesis Block Creation

Since all three nodes wrote the same genesis state, each node had a reply notification of successfully created block with the same hash as this all nodes are now under one blockchain network and ready to create a block for the same chain under the same network and each have the same initial genesis block.

After successfully creating the genesis block and syncing all three nodes we paired them with each other. Since in a decentralized network like blockchain all the nodes must be connected to each other to validate the transaction, update and sync the chain.

Figure 4.2 shows the information of two connected peers of node1. We see there are two different enode that represent two other nodes currently existing in the network.
 To show all the necessary transaction details we used ethereum network stats which

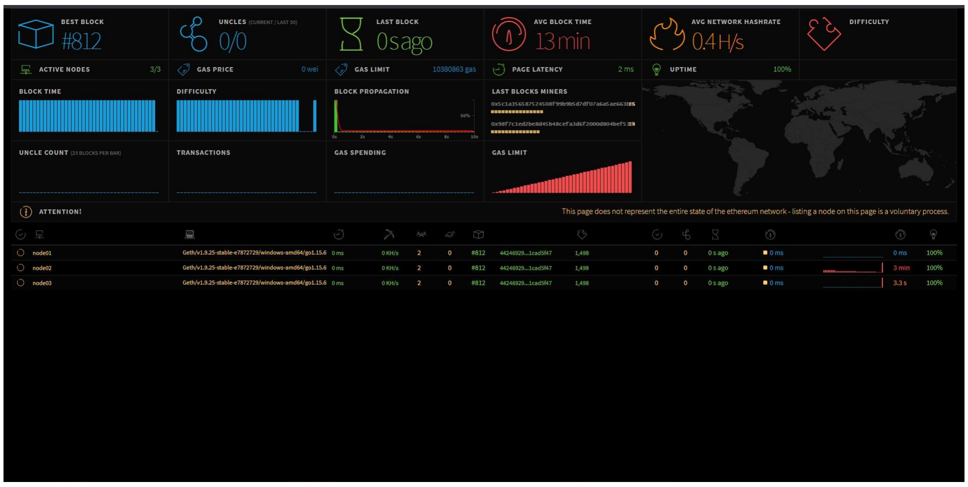


```
> admin.peers
[{
    enode: "enode://7c6e00595a919c21b6570d3fb845912e8a0a77fe61f02696683871767aa2
b235c5755290c3e402c4abc82ad58cd35ec271c035c8991dd35cbf2dfbcb8b517a62@127.0.0.1:3
0312",
    name: "Geth/v1.9.25-stable-e7872729/windows-amd64/go1.15.6",
    network: {
      inbound: false,
      localAddress: "127.0.0.1:57261",
      remoteAddress: "127.0.0.1:30312",
      static: true,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 545,
        head: "0x0e0111e1cc56a7614aa69af87ca8c3a39a98b9adf362c4f9d6e4bf4bcca4649
a",
        version: 65
      }
    }
}, {
    caps: ["eth/63", "eth/64", "eth/65"],
```

```
}, {
    caps: ["eth/63", "eth/64", "eth/65"],
    enode: "enode://3efe6a2a765b81d16b8e7a351d6db6fd6bfa3ee4678a1d890bfc220743de
f23c93e44d6817449e0882775fd152e0a6f45c950449cb6ce33aa89e62f84f940ce3@127.0.0.1:3
0311",
    name: "Geth/v1.9.25-stable-e7872729/windows-amd64/go1.15.6",
    network: {
      inbound: false,
      localAddress: "127.0.0.1:59715",
      remoteAddress: "127.0.0.1:30311",
      static: true,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 525,
        head: "0xe687e7873c8a6867a72c21ccd6895e9d7f6ba49a3b174f4327cb63be13366a2
2",
        version: 65
```

Figure 4.2: Connected peers in a single node

works as a blockchain explorer in this case. This allows users to track all the public information to explore.
 This will show almost all the necessary public information to explore such as av-



BEST BLOCK #812
UNCLES (CURRENT / LAST 50) 0/0
LAST BLOCK 0s ago
AVG BLOCK TIME 13min
AVG NETWORK HASHRATE 0.4H/s
DIFFICULTY

ACTIVE NODES 3/3
GAS PRICE 0 wei
GAS LIMIT 10380863 gas
PAGE LATENCY 2 ms
UPTIME 100%

LAST BLOCKS MINERS
0x5c1a35658752450 8f99b9b5d7df07a6a5ae663b 15
0x98f7c1ed2be8d45b48cefa3d6f2000d804bef53 14

ATTENTION! This page does not represent the entire state of the ethereum network - listing a node on this page is a voluntary process.

node01	Geth/v1.9.25-stable-e7872729/windows-amd64/go1.15.6	0 ms	0 KH/s	2	0	#812	44246929...1cad5f47	1,498	0	0	0 s ago	0 ms	0 ms	100%
node02	Geth/v1.9.25-stable-e7872729/windows-amd64/go1.15.6	0 ms	0 KH/s	2	0	#812	44246929...1cad5f47	1,498	0	0	0 s ago	0 ms	3 min	100%
node03	Geth/v1.9.25-stable-e7872729/windows-amd64/go1.15.6	0 ms	0 KH/s	2	0	#812	44246929...1cad5f47	1,498	0	0	0 s ago	0 ms	3.3 s	100%

Figure 4.3: Ethereum Network Stats

erage block time, gas price, transaction, validator numbers, average network hash rate, difficulty status, etc. These will help users to understand to set gas bid rates and miners to mine.

All these were done using git bash. The command that we used for creating these geth ethereum network and network stats are given below.

Creation of Validators nodes :
For creation each of the node needs to set a password also.
Node1:

```
    geth --datadir node1/ account new
```

Node2:

```
    geth --datadir node2/ account new
```

Node3:

```
   geth --datadir node3/ account new
```

After creating these three nodes, we created the genesis.json file using the CLI wizard puppeth.

Run Node1:
geth –nousb –datadir=*pwd –syncmode*
‘*ful*′ − −*port* 30310 − −*miner.gasprice* 0 − −*miner.gastarget* 470000000000 − −*http* − −*http.addr*‘*localhost*′
− −*http.port* 8545 − −*http.api admin*, *eth*, *miner*, *net*, *txpool*, *personal*, *web*3 − −*mine* − −*allow* − *insecure* − *unlock* –*unlock**PublicKeyof*
*node*1” − −*password password.txt*

Run Node2:
geth –nousb –datadir=*pwd –syncmode*‘*ful*′ − −*port* 30311 − −*miner.gasprice* 0 − −*miner.gastarget* 470000000000 −−*http* −−*http.addr*‘*localhost*′−−*http.port* 8546 − −*http.api admin*, *eth*, *miner*, *net*, *txpool*, *personal*, *web*3 −−*mine* −−*allow*−*insecure*−*unlock* –*unlock**PublicKeyofnode*2” − −*password password.txt* − −*ipcdisable*

Run Node3:
geth –nousb –datadir=*pwd –syncmode*‘*ful*′ − −*port* 30312 − −*miner.gasprice* 0 − −*miner.gastarget* 470000000000 −−*http* −−*http.addr*‘*localhost*′−−*http.port* 8547 − −*http.api admin*, *eth*, *miner*, *net*, *txpool*, *personal*, *web*3 −−*mine* −−*allow*−*insecure*−*unlock* –*unlock**PublicKeyofnode*3” − −*password password.txt* − −*ipcdisable*

Each node will return a enode after getting started and this needs to be save safely for future.

Using Geth Javascript Console:

To use the private geth network and interact with the blockchain we need to use javascript console of geth. For this we need to use the following command in separate terminal of the CLI.

```
geth attach http://:localhost:port
```

Here port number is different for each of the node of the network.

Peer Connection between nodes:

After running the js console it is time to interect with the nodes inside network. To connect nodes we need enode of each of the node that were previously generated at the time of running the nodes. To connect node with each other one node needs the following command incluing the enode of the node to which it is going to be connected.

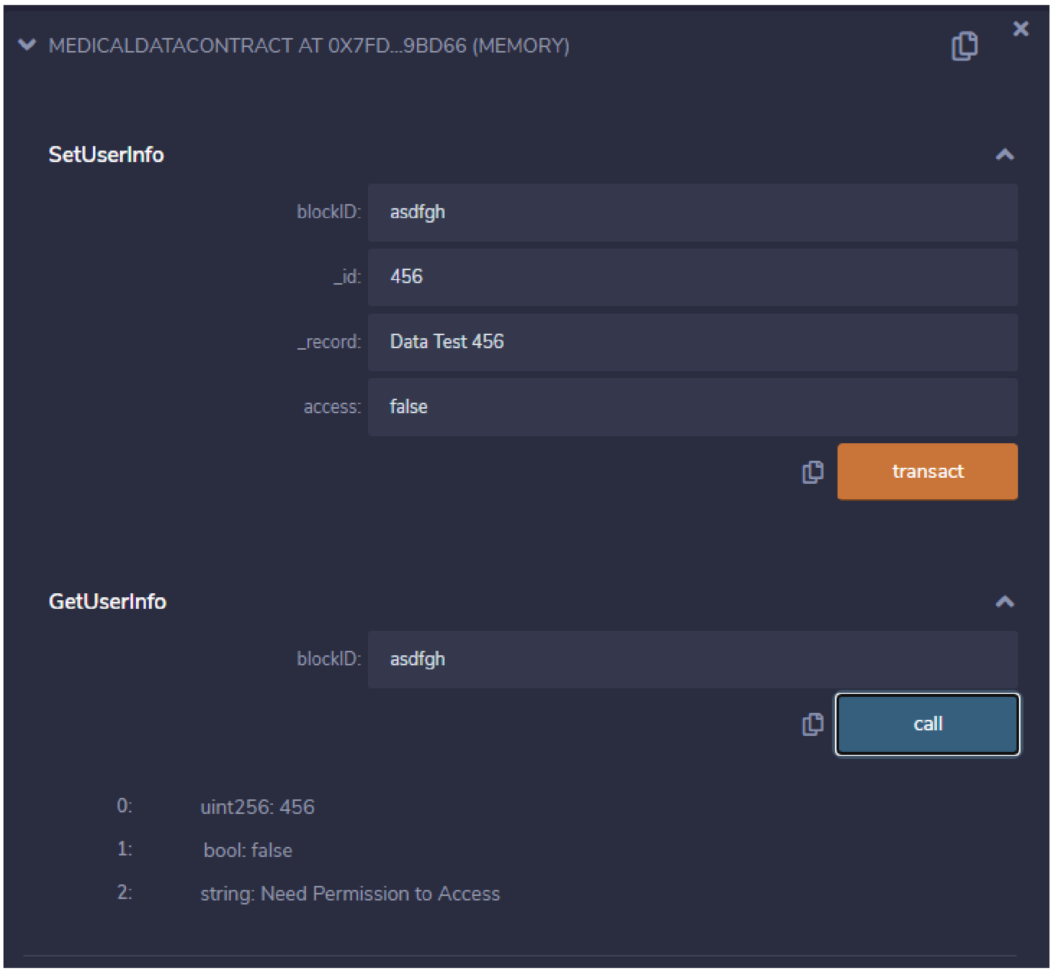
```
    admin.addPeer("enode of the particular node to be connected")
```

This command will build a synchronous connection between the two nodes.

4.2.2 Smart Contract

Apart from these private network creation and block synchronization to a single blockchain, we created a smart contract which has some small feature of our proposed model. For this task, we used the browser version of "Remix IDE" designed for the creation of a blockchain smart contract. For the creation of this smart contact, we used solidity as our programming language. Currently, there are a lot of versions for the solidity compiler. However, any of those compilers higher than version 0.5.0 wil lsupport our created smart contract.On that smart contract, we had provided three data fields to store data which were user ID, the main medical records and access permission. Here access permission is used as a boolean variable for testing purposes.

However, In future, this access mechanism will totally be changed according to our proposed model. By then, we manually set access permission simply by using boolean value as true or false.


MEDICALDATACONTRACT AT 0X7FD...9BD66 (MEMORY)

SetUserInfo

blockID: asdfgh

_id: 456

_record: Data Test 456

access: false

transact

GetUserInfo

blockID: asdfgh

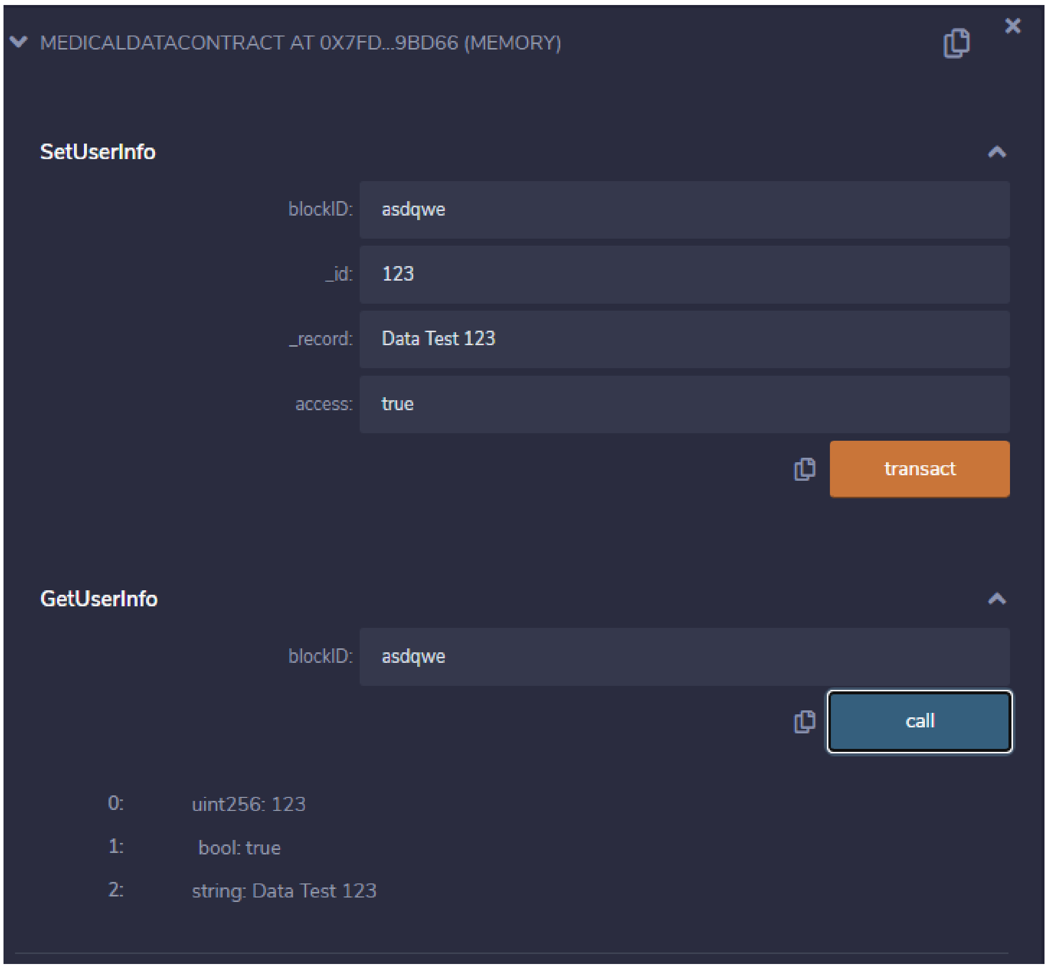
call

```
0:   uint256: 456
1:   bool: false
2:   string: Need Permission to Access
```

Figure 4.4: Data Store and Access Denied in blockchain in Remix IDE

If it is set as the true user will get access to blockchain data otherwise access request will be rejected.However, In future, this access mechanism will totally be changed according to our proposed model. By then, we manually set access permission simply by using boolean value as true or false. If it is set as the true user will get access to blockchain data otherwise access request will be rejected. Figure 4.4 shows that there is a “SetUserInfo” section where we user can put the data and click transact to store the data inside a block. In the section named “GetUserInfo”, if we write that particular block ID and press call, we see that no medical data is retrieved since initially the boolean value of access permission was set as false. Thus the user gets a message saying “Need Permission to Access”.

Similarly, figure 4.5 shows another transaction, However in this case for accessing data, the request is getting approved since while storing the data, the user sets the access variable as true. This is the way that our proposed model will work.



MEDICALDATACONTRACT AT 0X7FD...9BD66 (MEMORY)

SetUserInfo

blockID: asdqwe
_id: 123
_record: Data Test 123
access: true

GetUserInfo

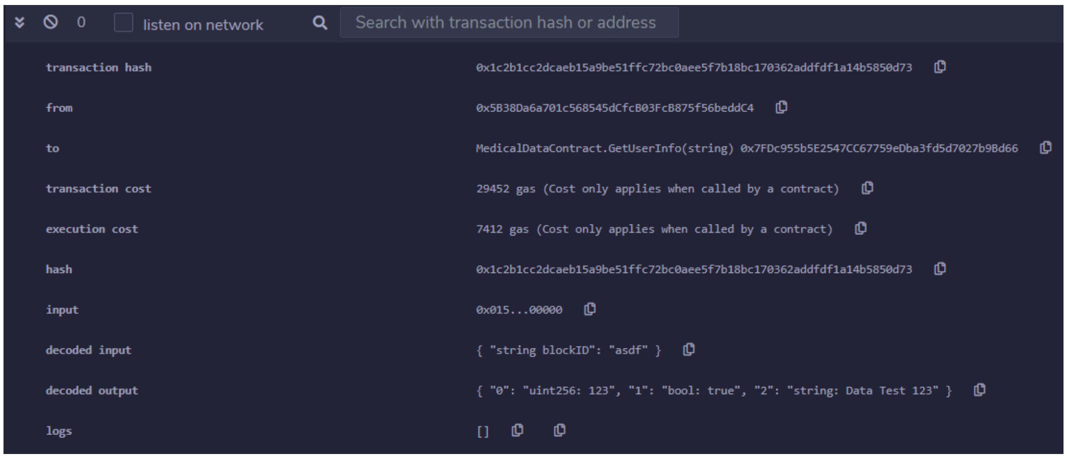
blockID: asdqwe

```
0:    uint256: 123
1:     bool: true
2:    string: Data Test 123
```

Figure 4.5: Data Store and Access Granted in blockchain in Remix IDE

However the access mechanism will be much more complicated as this access mechanism will check for access keys, signatures, hash values, etc. that are mentioned in the proposed model. The block will grant access to data if and only if that pieces of information are strictly checked and verified. Remix IDE has a good feature which is, we can show the transaction information inside it. However, this is not the real transaction but a test one that is similar to the real transaction.

Figure 4.6 shows the transaction details shown by the IDE after a particular completion of the transaction.



```
transaction hash    0x1c2b1cc2dcaeb15a9be51ffc72bc0aee5f7b18bc170362addfdf1a14b5850d73
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  MedicalDataContract.GetUserInfo(string) 0x7FDc955b5E2547CC67759eDba3fd5d7027b9Bd66
transaction cost    29452 gas (Cost only applies when called by a contract)
execution cost      7412 gas (Cost only applies when called by a contract)
hash                0x1c2b1cc2dcaeb15a9be51ffc72bc0aee5f7b18bc170362addfdf1a14b5850d73
input               0x015...00000
decoded input       { "string blockID": "asdf" }
decoded output      { "0": "uint256: 123", "1": "bool: true", "2": "string: Data Test 123" }
logs                []
```

Figure 4.6: Transaction report in Remix IDE

4.2.3 IPFS

To-Do this experiment we used the nodejs packages IPFS provides us with to use IPFS with JavaScript and make a web application. We have created a web app that allows us to upload a document, adds it to the IPFS server and then returns us a link that we can view in the browser.

To start working on creating the webpage we need to open git bash and run the following command

```
$ npm init
```

This Utility will create a package.json file that covers the most common items and tries to guess sensible defaults. After filling up the author name it will create a package.json file. Then using the following commends some necessary packages will be installed.

```
$ npm install ipfs-http-client ejs client ejs express-fileupload body parser
```

After the following packages are installed, we can start coding our app.js, home.ejs, and upload.ejs. We wrote the following codes.

app.js

```
const ipfsClient = require('ipfs-http-client');
const express = require('express');
const bodyParser = require('body-parser');
const fileUpload = require('express-fileupload');
const fs = require('fs');
const ipfs = new ipfsClient({ host: 'localhost', port: '5001', protocol: 'http'});
const app = express();

app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({extended: true}));
app.use(fileUpload());
app.get('/', (req, res) =>{
res.render('home');
});

app.post('/upload', (req, res) =>{
const file = req.files.file;
const fileName = req.body.fileName;
const filePath = 'files/' + fileName;

file.mv(filePath, async(err) =>{
if (err) {
console.log('Error: failed to download the file');
return res.status(500).send(err);
}


const fileHash = await addFile(fileName, filePath);
fs.unlink(filePath,(err) => {
if (err) console.log(err);

});

res.render('upload', {fileName, fileHash});
});
});

const addFile = async(fileName, filePath) =>{
const file = fs.readFileSync(filePath);
const fileAdded = await ipfs.add({path: filename, content: file});
const fileHash = fileAdded[0].hash;

return fileHash;
};

app.listen(3000, () => {
console.log('Server is listening on port 3000');
});
```

Home.ejs file

```
    <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name= "viewport" content= "width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compayible" content="ie=edge">
<title>Records</title>
</head>
<body>
<h1>Upload file to IPFS</h1>
<form action="/upload" method="POST" enctype="multipart/form-data">
<label>Filename</label>
<input type="text" name="fileName">
<br><br>
<label>Upload file</label>
<input type="file" name="file">
<br><br>
<input type ="submit" value ="Submit">
</form>
</body>
</html>
```

Upload.ejs file

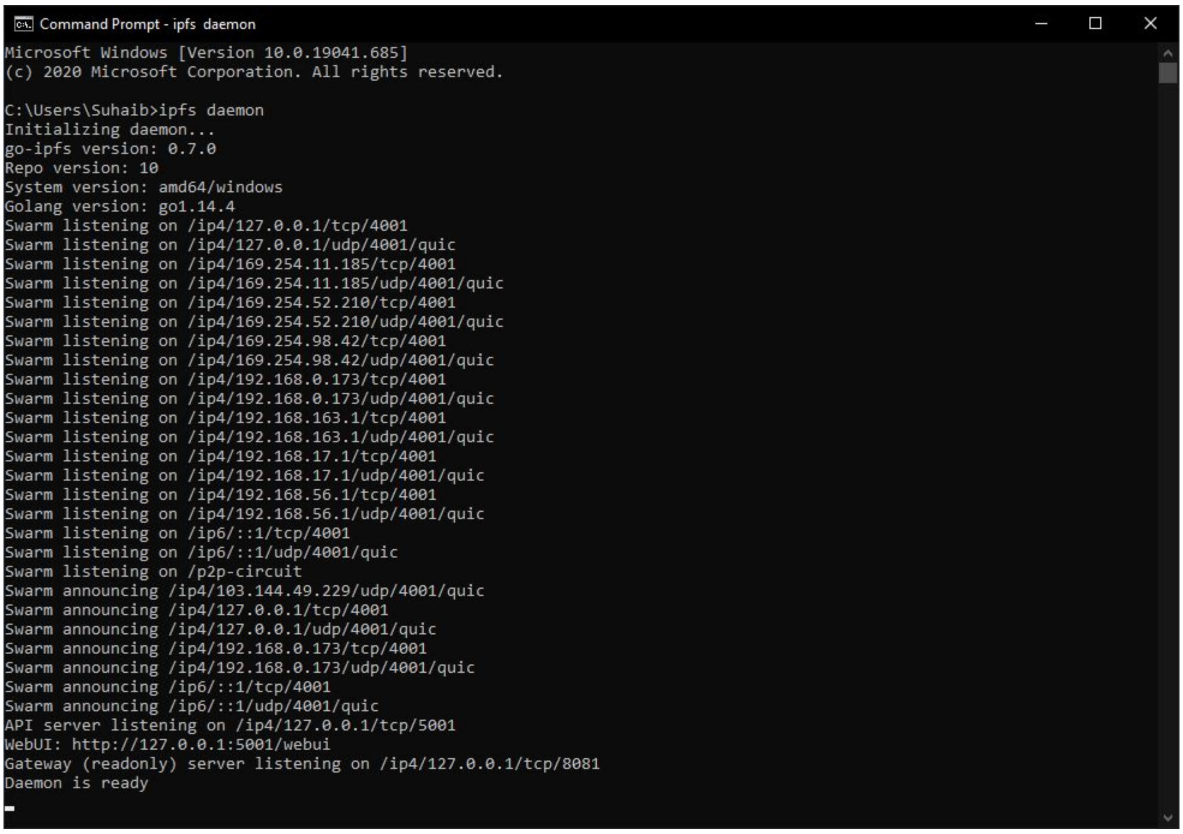
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name= "viewport" content= "width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compayible" content="ie=edge">
<title>Records</title>
</head>
<body>
<h1>File Uploaded!</h1>
<p>Name: <%= filename%></p>
<p>Link: <a href="https://ipfs.io/ipfs/<%= fileHash %>"><%= fileHash %></a></p>

</body>
</html>
```

After completing the coding part we can start the IPFS daemon server. We used the command prompt and wrote the following code.

```
Ipfs daemon
```

This command will run the server in port 5001. Here API server is listening on /ip4/127.0.0.1/tcp/5001. When the command prompt shows that the daemon is ready we can start the next part which is running the app.js part.



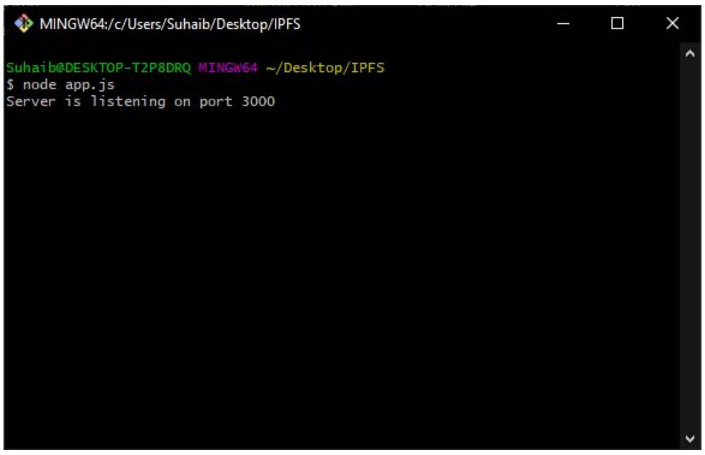
```
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Suhaib>ipfs daemon
Initializing daemon...
go-ipfs version: 0.7.0
Repo version: 10
System version: amd64/windows
Golang version: go1.14.4
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic
Swarm listening on /ip4/169.254.11.185/tcp/4001
Swarm listening on /ip4/169.254.11.185/udp/4001/quic
Swarm listening on /ip4/169.254.52.210/tcp/4001
Swarm listening on /ip4/169.254.52.210/udp/4001/quic
Swarm listening on /ip4/169.254.98.42/tcp/4001
Swarm listening on /ip4/169.254.98.42/udp/4001/quic
Swarm listening on /ip4/192.168.0.173/tcp/4001
Swarm listening on /ip4/192.168.0.173/udp/4001/quic
Swarm listening on /ip4/192.168.163.1/tcp/4001
Swarm listening on /ip4/192.168.163.1/udp/4001/quic
Swarm listening on /ip4/192.168.17.1/tcp/4001
Swarm listening on /ip4/192.168.17.1/udp/4001/quic
Swarm listening on /ip4/192.168.56.1/tcp/4001
Swarm listening on /ip4/192.168.56.1/udp/4001/quic
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/::1/udp/4001/quic
Swarm listening on /p2p-circuit
Swarm announcing /ip4/103.144.49.229/udp/4001/quic
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Swarm announcing /ip4/192.168.0.173/tcp/4001
Swarm announcing /ip4/192.168.0.173/udp/4001/quic
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/::1/udp/4001/quic
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8081
Daemon is ready
```

Figure 4.7: Running IPFS daemon server

After running the daemon server we can start the app.js. to start that we need to go to the directory where the app.js file is located and then open git bash. Here we used the following command.

```
$ node app.js
```

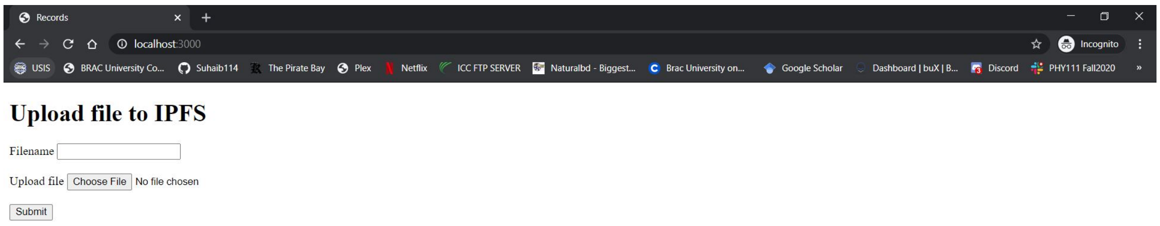


```
MINGW64:/c/Users/Suhaib/Desktop/IPFS

Suhaib@DESKTOP-T2P8DRQ MINGW64 ~/Desktop/IPFS
$ node app.js
Server is listening on port 3000
```

Figure 4.8: Running app.js

After running the server we can use any browser to access localhost:3000. Where we can see our working IPFS server. server.



Upload file to IPFS

Filename

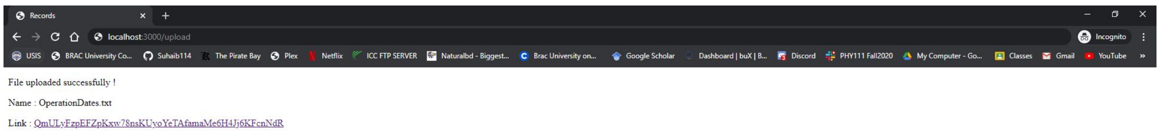
Upload file Choose File No file chosen

Submit

Figure 4.9: Upload files to IPFS

Here we can upload any types of files which will then store into the ipfs.We have uploaded a text file for reference named OperationDates.txt. After uploading a text file we get a confirmation dialogue and the hash value of that text file. This hash value is unique and will only for this file.

This way we can upload documents and image files in IPFS storage. Using this method we can also implement registration and store those data in storage as a text file.



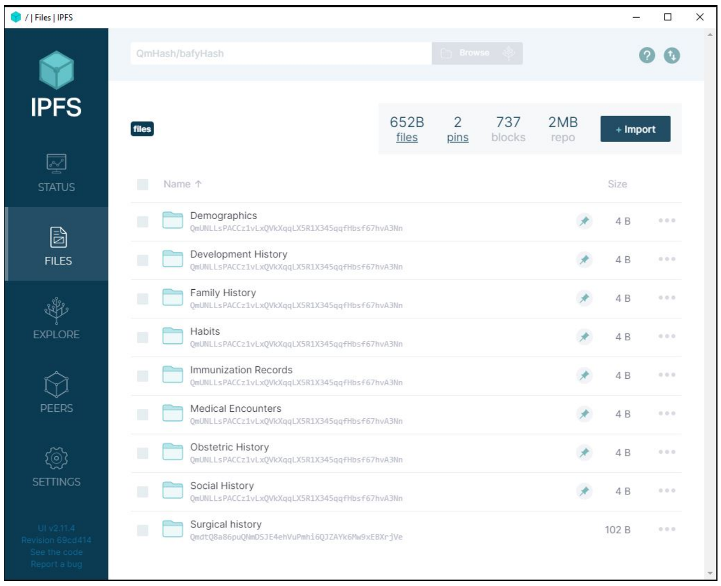
```
File uploaded successfully !

Name : OperationDates.txt

Link : QmULyFzpEFZpKxw78nsKUyoYeTAfamaMe6H4Jj6KFcnNdR
```

Figure 4.10: File Uploaded

We have also implemented the desktop version of IPFS to create a private server system. This method is more polished and more user-friendly. Here we categorized the directory for the reports so that they can be found easily. Here we have followed the traditional medical chart format so that physicians can easily understand them[S2].

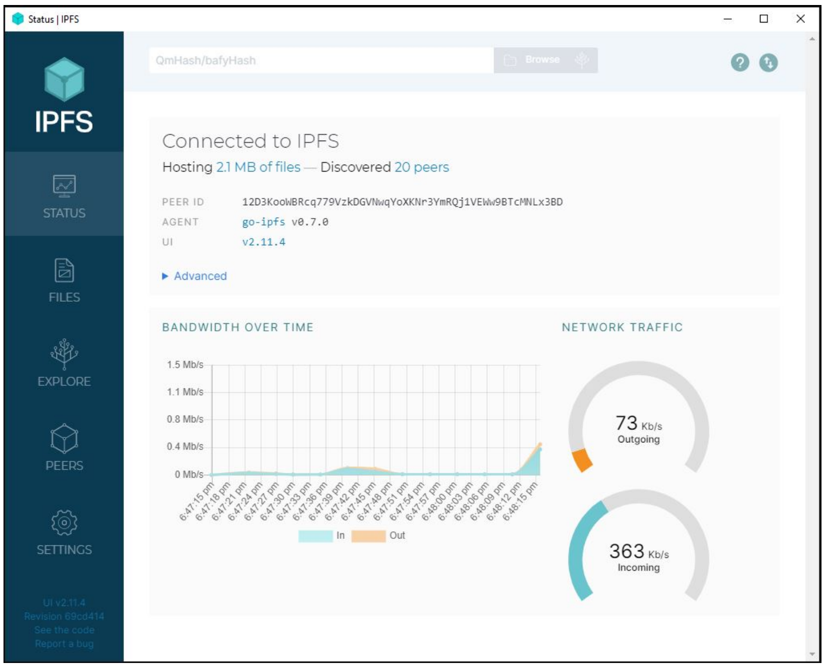


Name	Size
Demographics QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Development History QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Family History QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Habits QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Immunization Records QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Medical Encounters QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Obstetric History QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Social History QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn	4 B
Surgical history QmdtQ8a86puQNmDSJE4ehVuPmhi6QJZAYk6Mw9xEBXrjVe	102 B

Figure 4.11: IPFS Files Category

Here in the files, we can see how we have categorized the directory according to the medical chart. These directories will be in the newest modified order. So the physician and patient can easily find out which directories have been modified. The documents inside them will also be in newly modified order and will also be time stamped.

Here in desktop version status report is also shown where connected peers, hosting files, peers ID, and bandwidth over time are shown. This is helpful for the user to monitor their data speed and connected peers.



Connected to IPFS

Hosting 2.1 MB of files — Discovered 20 peers

```
PEER ID    12D3KooWBRcq779VzkDGVNwqYoXKNr3YmRQj1VEWw9BTcMNLx3BD
AGENT      go-ipfs v0.7.0
UI         v2.11.4
```

▸ Advanced

Figure 4.12: IPFS File Status

Chapter 5

Conclusion

5.1 Conclusion

To wrap up, our proposed model is a kind of EMR system which would be a patient centric with layers of security structured. In this, data is encoded in blockchain and IPFS cloud where other than the patient, no user can access the reports without permission. Physicians get the decryption key to encrypt the data only if the system can identify them authorized through CoAs and patients choose them to give the key which makes the system safe from third party interruptions. Sensitivity of data leakage especially when it comes to health history, has always been a major concernin every phase of life. As previously mentioned, there are numerous health data hacking incidents over the decades which led to focus on most possible secured EMR transfers. Needless to say, blockchain has created a great impact to build different featured protected systems till now. Our system is another upgraded addition to those which combines blockchain and decentralized IPFS cloud technology and assure that accountability stays only on the patient side. However, to set the seal, few operations are left to be carried out which we intend to do as our future work.

5.2 Future Work

According to our proposal, few actualization are needed to bring off the system as planned. Firstly, we are working on creating network with ethereum blockchain which needs to be connected with cloud. For that, smart contract is due to be completed with blockchain's data flow. As one of the main, connectivity of blockchain and cloud is yet to be completed, code of contracts need more development. Our thesis will meet a big fulfillment if these objectives are rounded off. Besides, till now we have done partial feasibility analysis so it needs to be taken care of in more details. Likewise,measuring the time complexity is required too. In order to increase smooth and fast data transmission rate, time complexity should be observed. It will help us to evaluate the process time and identify the need of new algorithm formulation. Later on, we plan to work on our own decentralized data sharing platform which can be a great addition for the system. On the whole, our target is to keep improving the system as much user friendly as possible with the utmost security standard and scalability.

Bibliography

[1] Bryant, M. (2019, February 13). Data breaches compromised 15.1M patient records last year. Healthcare Dive. https://www.healthcaredive.com

[2] Anderson, J., & Rainie, L. (2018, April 17). The Future of Well-Being in a Tech-Saturated World. Pew Research Center: Internet, Science & Tech. https://www.pewresearch.org

[3] Chhanabhai, P. (2007b, January 11). Consumers are ready to accept the transition to online and electronic records if they can be assured of the security measures. PubMed. https://pubmed.ncbi.nlm.nih.gov

[4] Eitan Bremler, Vice President Products and Marketing, Safe-T. (2016, March 17). 10 Ways patient data is shared with hackers: Personal healthcare information is becoming more difficult to secure. Security hacks of electronic medical records more than doubled last year, costing the healthcare system $50 billion, according to the American Action Forum. An IDC Health Insights report predicts 1 in 3 health records will be. Beckers Hospital Review https://www.beckershospitalreview.com

[5] Bryant, M. (2019b, February 13). Data breaches compromised 15.1M patient records last year. Healthcare Dive. https://www.healthcaredive.com

[6] The importance Of Data Security - a blog by sales-i. (n.d.). Sales-I. https://www.sales-i.com

[7] Creative. (2020, May 5). The Biggest Healthcare Data Breaches in History and What You Can Learn From Them. Paranet Solutions. https://www.paranet.com

[8] Siegenthaler, K. (2016, September 30). A Brief History of the EMR. Extract Systems. https://www.extractsystems.com

[9] Rosario, C. (n.d.). 4 Problems With Electronic Health Records. ADSC. https://www.adsc.com

[10] M. T. de Oliveira et al., "Towards a Blockchain-Based Secure Electronic Medical Record for Healthcare Applications" ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-6, doi: 10.1109/ICC.2019.8761307.

[11] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," in IEEE Access, vol. 6, pp. 53019-53033, 2018. [12] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, [online] Available: https://bitcoin.org/bitcoin.pdf.

[13] V. Buterin, Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform, 2013, [online] Available: https://github.com/ethereum/wiki/wiki/White-Paper.

[14] N. Álvarez-Díaz, J. Herrera-Joancomartí and P. Caballero-Gil, "Smart contracts based on blockchain for logistics management", Proc. 1st Int. Conf. Internet Things Mach. Learn., pp. 73:1-73:8, 2017.

[15] C. Dannen, Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners, 2017.

[16] S. Cheng and S. Lin, "Mining Strategies for Completing the Longest Blockchain," in IEEE Access, vol. 7, pp. 173935-173943, 2019.

[17] Q. Zheng, Y. Li, P. Chen and X. Dong, "An Innovative IPFS-Based Storage Model for Blockchain," 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Santiago, 2018, pp. 704-708, doi: 10.1109/WI.2018.000-8.

[18] How IPFS works. (2020, November 27). IPFS Docs. https://docs.ipfs.io

[19] D. C. Nguyen, P. N. Pathirana, M. Ding and A. Seneviratne, "Blockchain for Secure EHRs Sharing of Mobile Cloud Based E-Health Systems," in IEEE Access, vol. 7, pp. 66792-66806, 2019, doi: 10.1109/ACCESS.2019.2917555.

[20] E. Nyaletey, R. M. Parizi, Q. Zhang and K. R. Choo, "BlockIPFS - Blockchain-Enabled Interplanetary File System for Forensic and Trusted Data Traceability," 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, pp. 18-25, doi: 10.1109/Blockchain.2019.00012.

[21] M. Hölbl, M. Kompara, A. Kamišalic, and L. N. Zlatolas, "A systematic review of the use of blockchain in healthcare,"Symmetry, vol. 10, no. 10,p. 470, 2018.

[22] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," IEEE Access, vol. 6, pp. 38437–38450, Jun. 2018.

[23] Yao M.,(2017) "Your Electronic Medical Records Could Be Worth $1000 To Hackers", https://www.forbes.com

[24] M. Tindera, "Government Data Says Millions Of Health Records Are Breached Every Year," Forbes, 25-Sep-2018. [Online]. Available: https://www.forbes.com

[25] Lord R., (2017) The Real Threat Of Identity Theft Is In Your Medical Records, Not Credit Cards, https://www.forbes.com

[26] G. Magyar, "Blockchain: Solving the privacy and research availability tradeoff for ehrdata: A new disruptive technology in health data management," in NC'17, Nov. 2017.

[27] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare,"IEEE Access, vol. 4, pp. 9239-9250, 2016.

[28] R. Guo, H. Shi, Q. Zhao, and D. Zheng, ”Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems,” IEEE Access, vol. 6, pp. 11676-11686,2018

[29]J.A.F. (1999, December 9). An Introduction to Public Key Infrastructure (PKI) [Slides]. Acsac. https://www.acsac.org/1999/papers/thu-c-1030-furlong.pdf

[30]Public Key Cryptography (PKC). (2011, September). Techopedia.Com. https://www.techopedia.com/definition/9021/public-key-cryptography-pkc#:%7E:text=Public%20key%20cryptography%20(PKC)%20is,private%20key%20may%20be%20u

[31]What is the Medical Chart? Records and History. (2015, December 18). Practice Fusion. https://www.practicefusion.com/medical-charts/#:%7E:text=A%20medical%20chart%20is%20a,and%20laboratory%20and%20test%20results.

[32]Public Key Cryptography (PKC). (2011, September). Techopedia.Com. https://www.techopedia.com/definition/9021/public-key-cryptography-pkc#:%7E:text=Public%20key%20cryptography%20(PKC)%20is,private%20key%20may%20be%20used.