



Privacy protecting blockchain for the Healthcare sector

Thesis Submitted in Partial Fulfilment of the
Requirements for the Degree of
MSc Big Data Technologies

Author: Christopher McGahon / S2102466

Supervisor: Professor Huaglory Tianfield

Date: 07/01/2022

Acknowledgement

A huge thanks to my mom for giving me the opportunities and experiences that have made me who I am and for taking care of me and keeping me well fed throughout this whole process. Finally, a huge thank you to my caring and loving girlfriend Emma, without her constant support and help this thesis would never have been completed.

Abstract

The healthcare sector at any given time is flooded with an influx of big data in the form of confidential medical records containing private patient information. The confidential records are constantly under threat from data breaches and attacks due to fact they are being managed using a centralized approach. This centralized approach to databases becomes a single point of attack for malicious users. This method of data storage is also typically limited by things like network connectivity faults, hardware issues, bottlenecking, and due to minimal data redundancy leads to data that is extremely hard to recover if it is lost unexpectedly. This approach also means that not all relevant information can be accessed by all users who need it when it comes to patient diagnoses due to the housing of data in one location. The current research outlines the privacy, immutability, and data handling concerns that arise because of the current approach and proposes Blockchain as a potential solution. Studies have been done into the use of blockchain to house data but none of these studies go into much detail regarding the coding and testing aspects of the technology but seem to be more grounded in theory. By carrying out literature and technology reviews, a framework was designed to test to see if Blockchain is not only secure but is able to uphold records of patients in a secure and private manner. The prototype healthcare based blockchain system architecture was thus created using a mixture of JavaScript, NodeJS, and Solidity for the purpose of this research. Several JavaScript libraries were also implemented to develop the prototype solution including Ecdjs, Crypto, and IPFS. The prototype, with the inclusion of encryption and cryptography techniques is used to test for the most common privacy related attacks and evaluate the performance and weaknesses of the system. A cost analysis is also carried out regarding the implementation of the proposed framework to test for the cost viability of the project. Results showed that overall performance of the blockchain excelled at what it was designed to do. It was also found to be secure against privacy breeches and hacks discovered within the research phase of the project. The result from the cost analysis showed that it was also economically viable. To conclude, issues of scalability and skill shortages are a concern regarding the implementation of the proposed solution. Blockchain based data storages does offer a solution

to privacy concerns, however, since the technology is still maturing it may still be yet to flourish.

Table of Contents

1 Introduction	9
1.1 Background of the research	9
1.2 Motivation.....	12
1.3 Research Question & Objectives.....	13
2 Literature Review	14
2.2 Data Sharing & Management.....	16
2.3 Data Privacy	16
2.4 Data Protection	17
2.5 Data handling	17
2.6 Traditional systems vs Blockchain.....	18
2.7 Issues & Vulnerabilities	18
2.7.1 51% Attacks.....	19
2.7.2 Interoperability	19
2.7.3 Cost analysis.....	19
2.7.4 Skill issues.....	20
2.7.5 Sybil Attack.....	20
2.8 Discussion.....	20
3 Technology Review	21
3.1 Blockchains components	21
3.1.1 The block	21
3.1.2 Ledger.....	Error! Bookmark not defined.
3.1.3 Merkle Tree Root	Error! Bookmark not defined.
3.1.4 Consensus Protocols	22
3.2 Ethereum & Smart Contracts	24
3.2 Taxonomy of blockchain systems	29
3.2.1 Permissionless / Public.....	30
3.2.2 Permissioned.....	30
3.3 Blockchain Storage Solutions (Off-chain Vs On-chain)	30
3.4 Data Encryption & Cryptography	33

3.4.1 Broadcast Encryption	34
3.4.2 Identity-Based Encryption.....	34
3.4.3 Attribute-Based Encryption	34
3.4.4 Search on Encrypted Data.....	35
3.5 Performance	35
4 Methods & System Design	36
4.1 Design Goals.....	36
4.2 Proposed Architecture	37
4.3 Different Stakeholders	38
4.3.1 Admin users	38
4.3.2 Doctors.....	38
4.3.3 Patients	38
4.4 Technologies used.....	38
4.4.1 Ethereum	38
4.4.2 Web3.js	39
4.4.3 Next.js & React.....	39
4.4.4 Meta Mask	39
4.4.5 IPFS.....	40
4.4.6 Solidity.....	40
4.5 Implementation	40
4.6 Functions.....	41
4.6.1 Registering Patients on the system.....	41
4.6.2 Signing up & Login of Users	41
4.6.3 Uploading data.....	42
4.6.4 Viewing and receiving data	42
4.6.5 Giving access to data.....	42
4.7 Front-End Features	44
4.7.1Administrative Dashboard	44
4.7.2 Doctor Dashboard	44
4.7.3 Patient Dashboard	45
4.8 Back-End using Smart Contracts	45
4.8.1 File contract	45

4.8.2 Admin User Contract.....	46
4.8.3 Doctor Contract.....	46
4.8.4 Patient Contract.....	47
4.8.5 Certificate Contract.....	48
4.8.6 Interaction across various smart contracts.....	49
5 User testing Results & Discussion	50
5.1 Privacy & Security Analysis	52
5.1.1 Injection	53
5.1.2 Broken Authentication & session management	53
5.1.3 Sensitive Data leaks.....	53
5.1.4 Broken Access Control	53
5.1.5 XML External entity.....	54
5.1.6 Security Misconfiguration	54
5.1.7 Cross-Site scripting.....	54
5.1.8 Insecure deserialization	54
5.1.9 Lack of monitoring and logging.....	55
5.1.10 Using components with known Vulnerabilities	55
5.2 Smart contract-based privacy concerns.....	55
5.2.1 DOA attacks.....	55
5.2.2 Overflow & Underflow	55
5.2.3 Parity multi-sig wallet attacks	56
5.2.4 Rubixi Attack	56
5.2.5 Short Address Attack.....	56
5.3 Computation of Costs	56
5.4 Execution Times	58
5.4.1 Execution Time to Upload a Record.....	Error! Bookmark not defined.
5.5 Scalability of the Prototype.....	59
5.6 Access Control Performance.....	61
5.7 Security Performance.....	62
5.7.1 Scenario 1.....	63
5.7.2 Scenario 2:.....	63
5.8 Issues Found.....	63

5.8.1 Scalability	64
5.8.2 Low Latency Networks	64
5.8.3 Data Uploading Issues	64
5.9 Results	65
6 Discussions	65
7 Conclusion & Future Work	66
9 References	69
10 Appendices	83
Appendix 1. List of Notations	83
Appendix 2. Elliptic-curve cryptography (ECC)	84
Encryption & Decryption	85
Signature Creation & Verification	86
Appendix 3. Elliptic Curve Qu Vanstone (ECQV)	86
Appendix 4. Source Code	88

List of Figures

Figure 1 The popularity of blockchain over the last few years (Farouk et al., 2020).....	11
Figure 2 The components involved in an EHR.	14
Figure 3 EHR network involving the patient	15
Figure 4 Pro's & Cons of Blockchain in Healthcare	21
Figure 5 Blockchain block architecture	22
Figure 6 Visualization of the ledger (Velmovitsky et al., 2021)	Error! Bookmark not defined.
Figure 7 Merkle Tree structure (Wikipedia., 2019).....	Error! Bookmark not defined.
Figure 8 Ethereum Transaction Structure (Lv et al., 2019)	28
Figure 9 Characteristics of Permissioned vs Permissionless Taxonomy (Xu et al., 2017).....	30
Figure 10 Off-Chain blockchain(Onik et al., 2019)	31
Figure 11 A GDPR compliant blockchain solution (Onik et al., 2019)	32
Figure 12 Proposed architecture	37
Figure 13 Variables & functions of the File Contract	Error! Bookmark not defined.
Figure 14 Variables & functions of the Admin User Contract.....	Error! Bookmark not defined.
Figure 15 Variables & functions of the Doctor Contract.....	Error! Bookmark not defined.
Figure 16 Variables & functions of the Patient Contract	Error! Bookmark not defined.
Figure 17 Variable & functions of the Certificate Contract	Error! Bookmark not defined.
Figure 18 Interaction between the different smart contract	50
Figure 19 Set-up for desired testing	51
Figure 20 Execution time for the uploading of records, performed by a doctor user..	Error! Bookmark not defined.
Figure 21 The result of asynchronous scalability testing	60

List of Tables

Table 1 Deployment costs (Gas) involved with the smart contracts	57
Table 2The cost of the core functions within the contracts	58

1 Introduction

1.1 Background of the research

Currently a person's medical records can be recorded in one of a few different mediums. These can be, electronically, on paper, or a hybrid of the two. This causes issues when it comes to diagnosing patients, as their records are stored on different mediums thus making it difficult to quickly and coherently gather their information to form a diagnosis. This, in conjunction with a fragmentation of services such as primary, secondary, or tertiary care, with the use of different systems makes it hard to communicate information, leading to miscommunication or missing information regarding patients (McAvey, 2021). Blockchain, a new, secure, and innovative system of recording information has been proposed as a potential solution to these various medical record issues. Blockchain as a concept is an interesting and acclaimed technology that promises to bring in a new era of the internet not seen since its own inception in 1983. Since its development Blockchain has been analyzed and portrayed in a very positive light, with many of its strengths being highlighted throughout the research field. In a survey done by the 'World Economic Forum', released in September 2015 it was stated that by the year 2025, up to 10% of the world's global gross domestic product will be stored using the technology (World Economic Forum, 2021). Due to the COVID-19 pandemic the healthcare sector saw a huge influx of patients, thus far greater demands were placed on patient record storage systems, many of which used methods which had not yet been modernized such as paper trails. Increased concern arises regarding how the medical records should be more securely stored and formatted. Major ethical and legal issues for both medical practitioners and their patients could arise if the medical records were not managed correctly (Tandon et al., 2020). Blockchain has been highlighted in several studies as a potential solution for these issues. Blockchain has proven to be an extremely secure system which can be evidenced by its success and secure use for Bitcoin. Blockchain, was initially set up for use with Bitcoin, which has taken the world by storm in recent years. It was developed by an anonymous user by the pseudonym 'Satoshi Nakamoto', who described blockchain initially in 2008 in a whitepaper as a solution for the issues related to ownership in the digital world (Satoshi, 2008). Blockchain in essence, is a combination of two older technologies, peer to peer communications and cryptography (Anderson, 2018). The bitcoin blockchain works by making

use of a decentralized collection of user wallets to process transactions. When an exchange happens from user A to user B, the information of this transaction is simultaneously shared with all other users on the blockchain. Information is combined (e.g., the sum of money being transferred, ownership of both users, timestamps, etc.) and is added as a new block on the bitcoin blockchain. Since it makes use of inbuilt cryptography trust systems, users do not need to have issues with trusting other users (Burniske et al., 2018). While this is happening, the exchange of data is verified and checked by all the other wallets on the chain. Due to this approach, blocks are added in chronological sequence and as so are completely anonymous and impervious to tampering (Dwyer, 2015; Böhme et al., 2015). This approach leads to the creation of the one single publicly available database that is still used by Bitcoin today. This approach also makes the Blockchain, completely anonymous, which is vital for medical records in healthcare. The decentralized approach it takes could also provide many advantages to the healthcare sector as it would mean that there would be far less chances of system failure, less chance of hacks and breaches of privacy. If blockchain was implemented in healthcare sector this would also mean that different medical departments and personnel could access important patient notes and information easily, and quickly without having to contact each other separately. This saves valuable professional time and money and means that all relevant patient information can be accessed at once. As well as this there are often many time-consuming tasks associated with patient records, such as filtering through past medical records, and requesting information from additional parties, thus the new blockchain system would make records more accessible (Swan, 2015).



Figure 1 The popularity of blockchain over the last few years (Farouk et al., 2020)

Blockchain makes use of global ledgers to store its data, this way any node on the network can access information. Behind each transaction, there is a person behind it whose personal data can be gathered. Generally, techniques such as encryption are used to solve these issues. All transactions are encrypted but to verify these transactions, they need to be shared with the whole network for the system to work. When using blockchain exchanges, users generally have a set address that transactions are sent to and from, like an ID or email address. The real identity of the user is anonymous however, if trends are established from a user a lot of information can be gathered, any data that can be used to identify a user within a set of peoples is described as personally identifiable information (PII) (Pfitzmann et al., 2010). That is why it is important to use proper privacy preserving techniques when it comes to this technology. There have been some uses of blockchain to combat these issues, however it is hard to discern what work has been done since there is little to no documentation. One such project is Medicalchain, created by Dr Abdullah Albeyatti and Mo Tayeb (Albeyatti et al., 2021). While this project shows promise, the roadmap for the project ends within the first quarter of 2021 with no updates, or any important information listed on their website.

1.2 Motivation

Data and information are very strong tools in our current world and healthcare related documents are exceptionally important. The abundance of wearable and application for phones has led to an overwhelming amount of information being generated. Blockchain based exchanges could offer various advantages in the way information is handled and stored (Swan, 2015). Blockchain is a distributed ledger system that can provide a potential solution for issues of privacy and data protection. The chronological nature of the technology allows information to be added block by block. This allows for step-by-step reviewing by medical experts. This technology also allows for data verification without a central authority to govern it and is done so by using public validation methods and consensus algorithms. Medical data is in theory stored by using hashes, the advantage this holds is that data is not able to be tampered with. Even if some users on the blockchain network are malicious its public nature allows it to remain operational without interruptions. Interoperability is an advantage, as the technology uses efficient scalable access to healthcare records across different parties. By making use of smart contracts via application interfaces (APIs) can possibly cut costs and time delays. For these reasons, blockchain continues to advance in other sectors such as the financial sector. Even in its baby stages, blockchain such as bitcoin have been investigated for their uses in the healthcare sector (Nakamoto, 2008). If used correctly blockchain could be applied to the healthcare sector. This would help to enhance the privacy and integrity of data. It also provides the following benefits (Shi et al., 2020).

- Pseudonymity: Using blockchain each user is 100% anonymous.
- Auditability: It is easy to keep an eye on data to comply to regulations and privacy acts.
- Trust: The decentralized approach means that there is less relying on third parties
- Autonomy: Users can control their own data by making use of smart contracts using blockchain.

Motivated by the advantages that blockchain may offer, within this study, it is proposed to research the state-of-the-art literature on offer and use the knowledge obtained to create a prototype blockchain solution for the healthcare fields. By making said prototype it is hoped that a foundation is built onto which testing around issues regarding privacy, security, and access

control can be carried out. By doing this a gauge of the feasibility of using the proposed system for real world application can be obtained.

1.3 Research Question & Objectives

There are several questions and objects that need to be carried out for this project to be deemed a success or not. The first of these is a comprehensive review of the literature available and deciding if there is a future for a blockchain based healthcare information exchange system or not. This would be done by viewing what work has already been done in the efforts of a more robust way of storing information. Since this technology is still rather new it is not clear or not if this will be a success. Secondly, based on the finding of the researching process an unbiased view and critic on using the technology for the exchange of information using Blockchain should be obtained, this will help find out if the technology is there and if it has any place in the future of the healthcare sector or if it is a passing fad.

Thirdly, the creation of a blockchain-based architecture should be carried out in the hopes of creating a secure way for doctors and patients to share medical records in a more secure and private manner. This is a steppingstone that is needed to be accomplished before we can hope to create a prototype to gain additional insights into the technology. This in-depth look and creation of a mock architecture will be used to get a hands-on approach to the problem that you cannot get from reading literature reviews. By the coding of a prototype and further utilization of a storage mechanism that would allow the ability to improve the scalability of the system proposed. The mentioned prototype should hopefully be in the form of the creation of a Decentralized Application (Dapp) by using the literature review as a guide as the best approach to take when carrying out this work. This prototype can finally be used to analyze the robustness of the developed Dapp against several security attacks most implemented against a data storing application such as the one proposed in this study. To conclude, the project proposes the creation of a blockchain based system that implements several modular parts to achieve the said goal of seeing if there is a future for blockchain in sectors such as healthcare where issues are bountiful surrounding ethics, privacy, and data breaches. This prototype will then be bombarded with attacks to test for its robustness in its

job to secure data, ultimately answering the question of ‘does blockchain have a future in the healthcare sector?’

2 Literature Review

There has been an ongoing shift from traditional paper trails to the implementation of systems using Electrical Health Records (EHR). This shift has led to an increased overall intelligence and quality when it comes to healthcare since it was first introduced in the 1970’s and enforced by the American government in 2009 (Atherton, 2011). This shift is said to have saved the sector more than billions yearly (Kemkarl et al., 2012). The shift to EHR has led to an increase in the quality and safety for all parties involved. EHRs are usually sent by physicians to laboratories and vice versa regarding test results (Jamshed et al., 2015). Fig. 2 shows the components involved with an EHR. Since there is a single pipeline to the server it is assumed that this may be used as a point of attack when it comes to malicious intent.

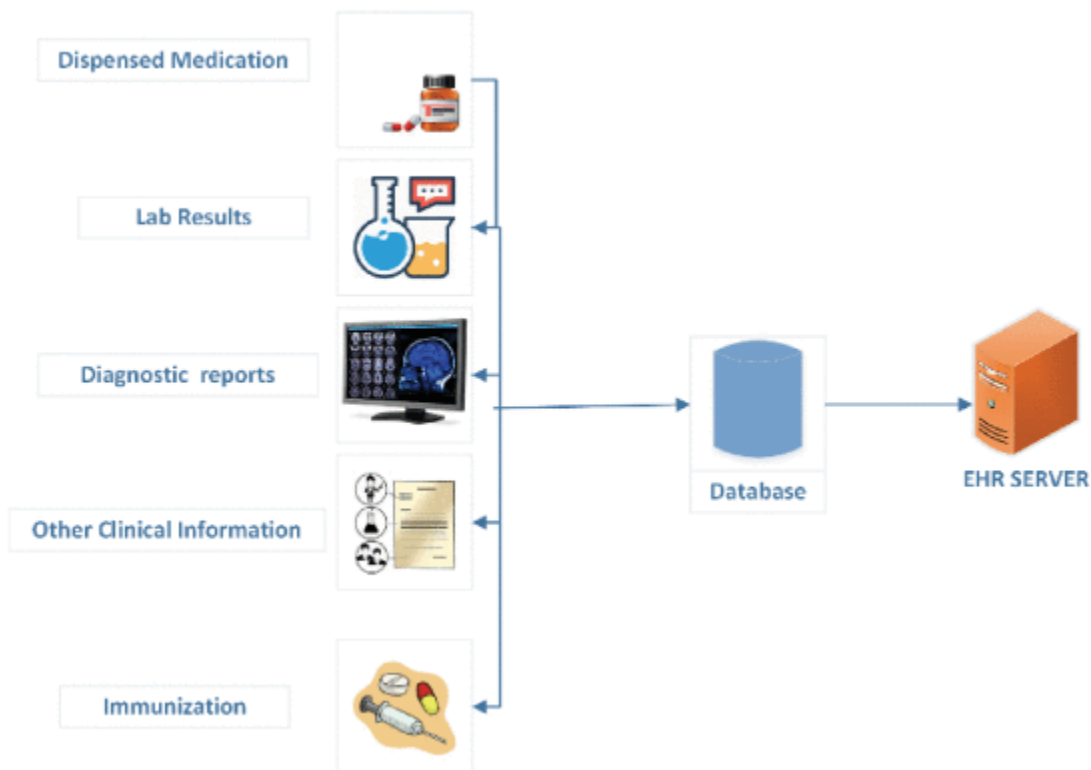


Figure 2 The components involved in an EHR.

Furthermore, fig. 3 shows the patient's involvement with said system. Since there is no direct link between the patient and the server housing the EHR's it is assumed that the patient has no direct control over what is done with said information.

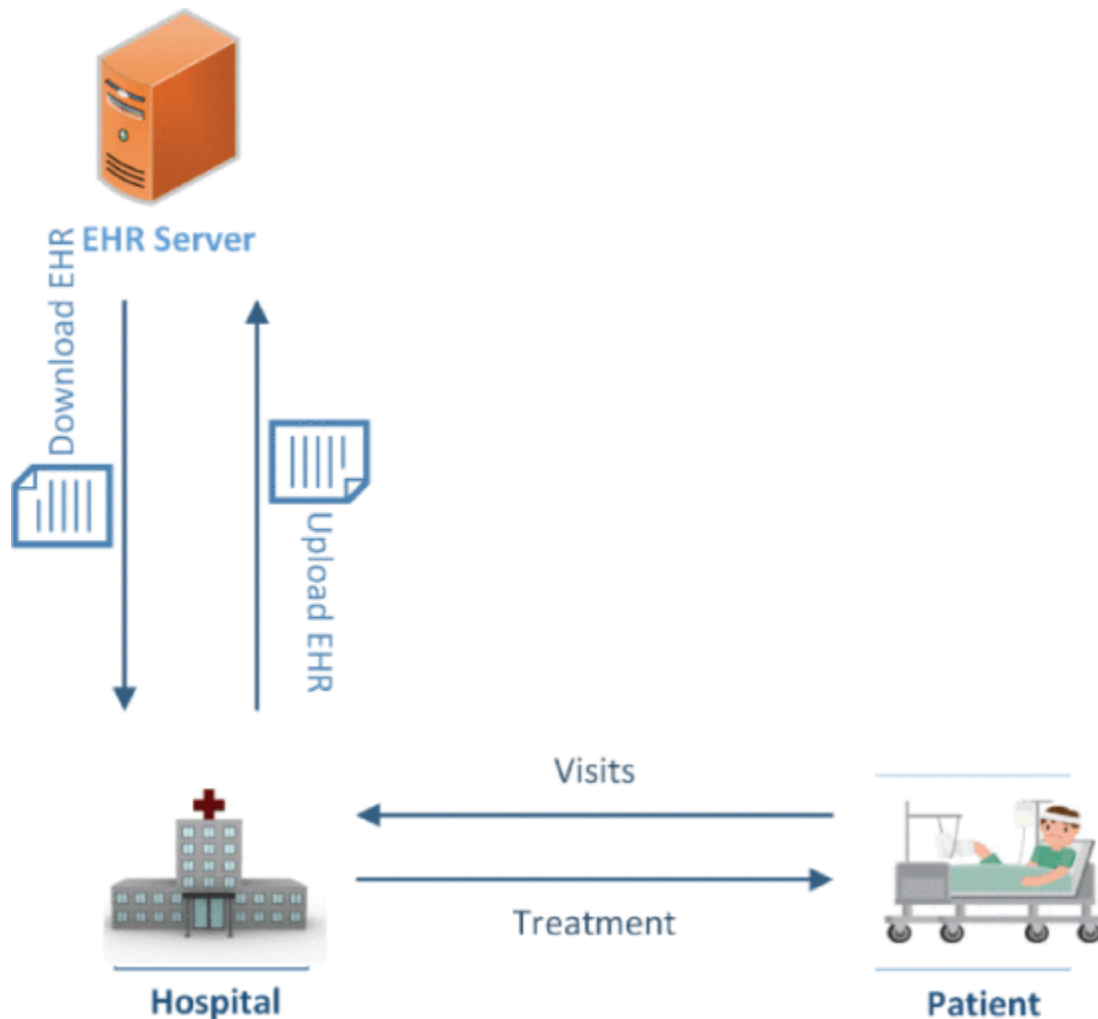


Figure 3 EHR network involving the patient

The issues of an EHR system are surrounding the involvement of the patients themselves who should own the data. However, under this system they generally can only read their data, having no say over who can access their information. True and total privacy would allow the user to decide who can see their data and when. Data gathered by a result of clinical interactions should give the patients whose data it is full confidentiality and privacy protection (Harman, 2001). However, there still are several issue revolving around this digitization, these are mainly seen in

the form of privacy concerns for users and patients. Reports show that a lack of proper treatment of records has led to an increase in data breaches over the last decade, leading to concerns of who is looking at our data and by extension what it is being used for (ONA General Counsel, 2016).

2.1 Data Sharing & Management

Blockchain could allow for patients to manage their own medical records, which could provide them with a greater sense of ownership over their own personal information. A study by Nguyen et al, (2019) found that patients who were left to manage their own private files without the involvement of a third-party reported having a greater level of confidence, autonomy, and control over their own records. This led to a higher level of satisfaction amongst the participants that had control over their own records in comparison to those who did not. Thus, the introduction of Blockchain to the healthcare sector could provide patients with a greater sense of autonomy over their files and a knowledge of where and who their personal files were being shared with. Based on the literature, the management of how medical records are handled seems to be where most of the research attention is pointed towards in the field. Overall, it was found that blockchain is an effective and more proficient method in comparison to more traditional models when it came to the management of medical data and EHRs (Quaini et al., 2018; Dagher et al., 2018). Furthermore, blockchain, in comparison to currently used centralized healthcare models, may offer a more feasible way of information exchange by incorporating more varied heterogenous types of data with higher variability as well as big data (Mamoshina et al., 2018).

2.2 Data Privacy

The protection of information by means of enforcing authorized access to the records has been a focus on the topic of blockchain. Research has shown that the management of controlled access has been a very important feature when it comes to the technology (Zheng et al., 2019). This issue is prevalent in the Healthcare sector especially due to the amount of sensitive data generated. In response to this, prior studies have been carried out regarding frameworks that allow a user centric, effective, and secure way for users to access patient records, as well as other key medical data (Guo et al., 2018; Badr et al., 2018; Dwivedi et al., 2019).

2.3 Data Protection

To help prevent unauthorized data access and to further assist on the security of data overall has been a key issue for the healthcare sector. Several studies have been focused around how to protect against these issues using blockchain. Some studies focused their attention on preventing the unauthorized access of data (Nguyen et al., 2019). Whereas other researchers focused on the issues of data eavesdropping (Uddin et al., 2018). However, a lot less focus has been aimed at the prevention of several external based attacks like collusion attacks (Guo et al., 2018). The transparent nature of the technology makes life simpler when it comes to tracking who can see records or transactions in the case of blockchain. Each block can be traced back to its inception. This allows for a more secure, transparent, and effective network solution when it comes to privacy (Tian, 2016).

2.4 Data handling

There is a greater need for a more ethical and legally sound way of gathering, sharing, and handling of healthcare data. Few studies mention the need for compliance or a standardized approach when it comes to the handling and managing said data (Omar et al., 2019). However, a focus of the sector is devoted to the overall need to maintain data integrity (Hyla & Pejaś, 2019; Shen et al., 2019). Research has also shown that there is an increased number of people accepting blockchain as a viable solution to issues regarding data handling (Dagher et al., 2018). Research has focused itself on how to approach the storage of sensitive data from different sources and has also shown a need for a better way to process data (Zhou et al., 2018). Studies have also suggested solutions to some of these issues. For example, on how to handle various data types (Quaini et al., 2018) and the benefits of incorporating smart contracts (Dagher et al., 2018). The main points addressed in these studies suggest that an emphasis on the improvement of handling features of blockchain, as well as the management of medical information make it a superior framework. They also noted that through the use of Blockchain data sharing was far more decentralized and inter-useable, making it a more accessible platform for patients and medical professionals. This thus shows that Blockchain has the potential to create a more stable user environment in regards both accessibility and user-privacy.

2.5 Traditional systems vs Blockchain

Traditional approaches for the handling of medical based data are formulated around the premise that the servers which house said data are trusted by all who own data. This extends to enabling the server to then control all access and authenticate on the information (Ying et al., 2018). However, these servers tend to be honest and curious at the same time this means that as it honestly performs requests, it may also leak information without its owner's consent. Since must networks make use of a predefined pointed of access I. e. a centralized server. This can lead to a point of failure regarding healthcare systems (Laxmeshwar, 2001).

Blockchain offers various features that could aid the healthcare sectors, such as access controls. Immutable ledgers are created to help keep records of information exchanges (Hölbl, 2018). These records of who or what is accessing the data cannot be tampered with or modified for ill gain. This helps to ensure a high level of integrity and trustworthiness. On top of this, by using blockchain to control access to users helps to promote transparency within the system. This solves for the issues surrounding data leakages. Smart contracts can be used (Wang et al., 2018) to help with both authentication as well as user verification. The enforcement of strict control / access protocols using smart contracts can also lead to the early detection of threats to the networks (Nguyen et al., 2019). Furthermore, blockchain may eliminate the reliance on the traditional centralized servers used by the healthcare sector, promoting fairness among those who use said systems. As smart contracts are typically public, meaning that all users can access them, this promotes the equality within a system. It also means that if any points of failure happen, they will not cause a loss of data and cause concerns surrounding trusts (Hölbl, 2018).

2.6 Issues & Vulnerabilities

Blockchain does not come without its own issues however, which are unique to how the technology's architecture and design are set up. The recent Health Service Executive (HSE) attacks in Ireland show how deadly and unprepared even a government body can be when securing confidential medical records (The Irish Times, 2021). Examples of such attacks that can happen using the technology are as follows.

2.6.1 51% Attacks

The data held on a blockchain are verified when most users validate said block. This is where 51% attacks come in. If malicious users were to obtain control of a majority of the nodes on a network, they may be able to attack the network. Since the cost of such actions is immense it is rare that attacks like this could happen (Kuo et al, 2017).

2.6.2 Interoperability

Healthcare data is information sensitive and generates huge volumes of data each day. Huge amounts of budgets are spent on tasks such as collecting, storing, and parsing through big data as neglect on these parts often leads to unsafe practices (HIQA, 2017). A Cultural change in the healthcare sector is required to ensure independent bodies are willing to share information. A standardization of the sector based around blockchain based exchanges is halted at the local level, meaning that there is a need for professionals to be educated on the benefits offered by blockchain (CHCF, 2005). Although the technology is used for cash transactions, companies such as IBM have been utilizing blockchain for tasks such as building supply chains. This transparency allows their customers to know when, where, or how products are being produced (IBM, 2018).

2.6.3 Cost analysis

Cost is a major constraint when it comes to blockchain. Reports have shown the cost estimation at serving files from 10 bytes all the way to up to 500 bytes were estimated to cost anywhere from €0.1 to €1.5 for both text and media formats such as imaging. The same report gathered data from 50 volunteering users and had them store data using a blockchain node. The estimating costs of doing so was an average of €200 per transaction using the Ethereum blockchain. The total data gathered being 259 kilo bytes (Franceschi, 2019). Additionally, the energy consumption associated with blockchain is large. The first algorithm to mine bitcoin called proof of work (PoW) was said to consume around 8 gigawatts of electricity per annum. This number rivaling that of countries such as Ireland (3 gigawatts per annum) and the United Kingdom (7.84 per annum) (Statista, 2021). Compared to other algorithms such as Proof of stake (PoS) which is used in the mining of ethereum uses nearly 90% less energy than Bitcoin (Digiconomist, 2021). Companies such as Deloitte have developed proof of concepts for systems that allow patients to own their own records completely. In this way suppliers can issue

prescriptions on the blockchain directly (Morris, 2016). In order to ensure the viability of the system, since it proposed to be used in a live healthcare scenario is the cost. As so the prototype will be designed in a manner and tested so that it also ensures the economic viability of itself. This will be done by carrying out costs analysis testing on the system once developed. In this way it can be gauged if the technology is useful in this sense.

2.6.4 Skill issues

The idea of blockchain is still a rather new and foreign idea for most people in the most industries. Switching from traditional infrastructure to blockchain would be a very time and cost consuming process. This would not only apply for hospitals but healthcare organizations (Alonso, 2019).

2.6.5 Sybil Attack

The issues of who is accessing user records is still prevalent when it comes to the block chain technology. In what is referred to as a 'Sybil' attack (Microsoft, 2002), a group of malicious users, posing as nodes, interfere and prevent the exchange of crucial user data. If data, such as hashed values of patient information are discovered it could compromise patient privacy. If physicians recognize keys belonging to a patient, they could use this to find out information they should not have access to (Alhadhrami et., al 2017). As the technology revolves around cryptographic algorithms, the introduction of quantum computers may put the whole technology at risk (Cloud Security Alliance, 2021).

2.7 Discussion

Based on the literature it seems that blockchain may answer the question of how medical records can be stored to promote privacy. However, it is not without its own faults. Fig. 4 outlines the keys strengths and weaknesses that the technology has based on the review. By making use of these findings the viability of the proposed system can be surmised when testing is carried out.

Strengths Secure Unable to tamper with records Makes health systems more reliable	Weaknesses Highly complex technology No 100% guarantee of data integrity High computation power needed
Benefits Patients manage their own data Reliable way of monitoring patients at home	Issues Privacy No current support for the technology in the sector Resistance to change from standard practices

Figure 4 Pro's & Cons of Blockchain in Healthcare

3 Technology Review

In this section a more in-depth look will be taken into the technologies involved with blockchain to gain an understanding of what is the best approach for the development of a system such as the proposed.

3.1 Blockchains components

Blockchain is a combination of technologies previously conceived. These are, distributed ledgers (DP), consensus protocols (CP), and cryptography (C). A blockchain can be formulated as the following: $Blockchain = \int(DL, CP, C)$ (Ismail et al., 2019). The components that make up a blockchain are.

3.1.1 The block

A The typical structure of a block in a blockchain can be seen below in fig. X (Zhang et al., 2019).

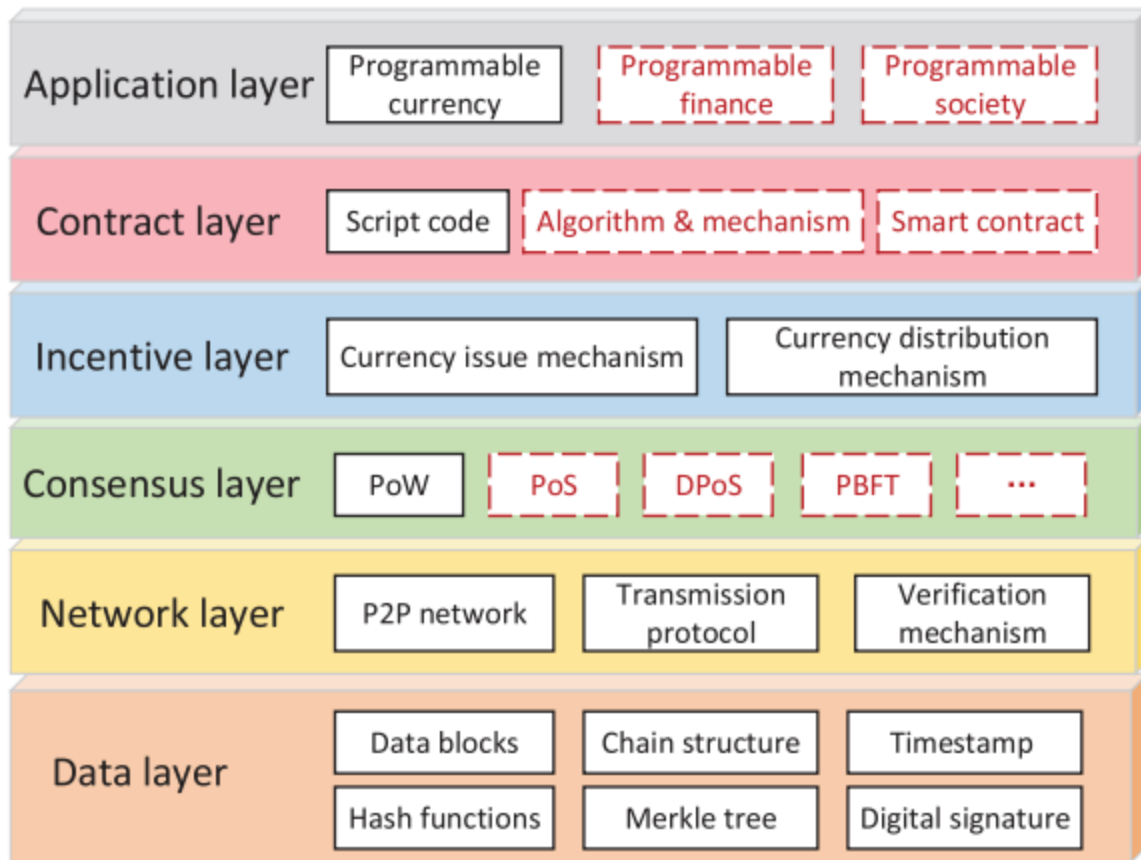


Figure 5 Blockchain block architecture

3.1.2 Consensus Protocols

The way that nodes on a given network verify exchanges is by use of a method referred to as a consensus protocol (Investopedia, 2021). The protocols or algorithms are used to decide how the exchanges are to take place. The most known examples of these protocols are PoW and PoS. Bitcoin, by making use of PoW implemented what is referred to as the Byzantine Fault Tolerance (BFT)(Hcl tech, 2021). BFT allows for two different nodes to safely exchange the same information even if up to 33% of the users on the network are malicious user's intent on causing issues (Lin et al., 2017). A more in-depth investigation into each consensus algorithm is as follows.

3.1.2.1 Proof of Work

PoW refers to when a block is being validated, every node works against each other to solve a hash to validate said block. This approach tends to be less about computational power and more about randomness. The nodes who compete are referred to as miners. The data they are trying

to validate is referred to as the 'Nonce' and is a fixed length random hash that is stored within the header of the block along with the 'difficulty' which is adjusted by the network to keep validation times level. When a miner thinks they have solved the problem, they share the answer with their peers. If most of the nodes agree that the answer is correct, then the miner who came up with the answer receives a reward. This reward is typically in the form of cryptocurrency, depending on the blockchain platform (Cachin & Vukolić, 2017). This approach incentivizes nodes to work honestly. Node groups make exchanges and select the blocks to be added to the chain. The more blocks a network has, the harder it is for a malicious user to pull off an attack. Since the probability of finding the Nonce n of proof H for a target is written as $P(H \leq T) = T/(2^{256})$. Since the target is a 256-bit number, the average attempts to figure it out can be represented as 2 to the power of the target. In this case it is 256 (Aitzhan & Svetinovic, 2018).

3.1.2.2 Proof of Stake

PoS is a newer algorithm than PoW and as so has been worked on with its issues in mind. These issues are related to cost and scalability concerns. The guess work is removed from validating blocks removing the need for powerful and specific hardware suited to mining. This helps to cut costs as less electricity and computing power is needed. This system works by making use of validator nodes. These nodes pay or 'stake' an incentive prior to work done to validate the exchange of information. Once all stakes are locked in a node is chosen at random and its hash is revealed to the other nodes on the network. The other nodes then bet on the validity of the block. If most of the validators bet on a single block, that block along with the users who bet on that block are rewarded from a pool of funds. If a majority consensus is not met on a block, that block's funds are then forfeited. This means that no computing is performed prior to selecting the chosen block. Chains such as Ethereum are examples of this system (Hcl tech., 2021).

3.1.2.3 Proof of Activity

Proof of activity (PoA) is a hybrid of both proofs talked about previously in which empty blocks are mined firstly by using PoW and then filled by making use of PoS to validate them (Investopedia, 2021).

3.1.2.4 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) is an extension of the Byzantine agreement protocol, which requires a set of parties in a distributed environment to come together and agree on a value, even if some of the parties involved are corrupt. Its origin goes all the way back to the Byzantine army (Castro & Liskov, 1999). Using BBFT, all nodes in the network are known. This limits the usage of consensus algorithms that can be used. There are three stages that the nodes need to move through to verify a block. These are pre-prepared, prepared, and finally commit. Every node needs to pass these stages to verify. Examples of this are seen in Hyperledger fabric (Hyperledger, 2021).

3.2 Choosing an overlying blockchain technology

To first build the proposed application using blockchain, a specific technology needs to be selected first. As so, it's important to not only learn what technologies are out there but what is best suited to our problem at hand. The market is full of plenty of able choices for blockchain, but mainly one that makes use of smart contracts narrows down the search. Some of the more popular choices (Liljeqvist, 2019) include Tezos, Electro-Optical System (EOS), Ethereum, Hyperledger Fabric, & Cardano.

3.2.1 Tezos

Tezos is a self-amending blockchain technology which makes use of on-chain mechanisms for the testing, selecting, proposing, and running of protocol updates without needing to make a radical change to the network protocol also referred to as a hard fork (Tezos, 2021). This allows for Tezos to be stable in the long run, anyone who wishes to build using Tezos can be ensured that the network won't suddenly drop, as seen historically with the likes of Blockchain & Ethereum. This self-amending nature of the technology has led to a large variety of car manufacturers to build using Tezos such as McLaren, BMW, Audi, and several others (Cryptonews, 2021). Besides this, the mentioned car companies have also praised Tezos for its ability to outperform Ethereum regarding the number of transactions per second the technology allows. Tezos is built using a functional programming language called Michelson, a low-level and stack-based programming language (Michelson, 2021). The functional aspect of the language allows for a mathematical validation of smart contracts on the chain, this means that unlike other

imperative programming languages which require a developer to specific instructions. SmartPy & Ligo are used to code smart contracts for Tezos. However, because of the way the ecosystem was developed many programmers that use Tezos are forced to create most of the solutions to problems themselves.

3.2.2 EOS

EOS (EOS, 2021) is the first of the blockchain based platforms to be researched. It is used as a decentralized medium to help host, develop, & is used to run decentralized applications. It was first introduced in June 2018 when its own cryptocurrency raised up to \$4 billion in revenue. EOS allows for developers to test out applications by holding their token rather than using them up during transactions as seen with other platforms. Often seen as direct competition with Ethereum, it promises to be bigger, faster, & better than Ethereum. While Ethereum can only handle around 15 transactions every second, EOS is trying to reach speeds of millions per second. However, as of writing EOS has not been able to achieve these speeds yet (Bit degree, 2021). Development of applications using EOS have seen a massive decline since its inception in 2018, with weekly coding updates dropping by over 90% in the latter half of 2020 with fewer developers using the platform overall (La Capra, 2021).

3.2.3 Cardano

Cardano is a blockchain platform developed by the co-founder of Ethereum Charles Hoskinson (Cardano, 2021). It was Initially developed around 2015 and then released in 2017 alongside it's token named ADA named after Ada Lovelace, who many consider to be the first computer programmer (Cardano, 2021). Billed as an alternative to Ethereum, Cardano sells itself as an upgrade to Ethereum, while using similar technologies such as smart contractors. As of October 2021, ADA is the fourth largest cryptocurrency in terms of its market value of \$69 billion (CoinMarketCap, 2021). However, Ethereum remains at the top of the food chain with almost seven times the value of Cardano (CoinMarketCap, 2021). In terms of its smart contracts, Cardano makes use of the Unspent transaction output accounting model (eUTXO). This is like the technology that Bitcoin uses, but with the added benefit of use of smart contracts which bitcoin it also makes use of a multi-asset support system which is the principle which cash works in the modern world (Sanchez, 2021). Ethereum, makes use of a system that revolves around an

account / balance system, where a user's coins are stored within an account, like how banking works (Mahut, 2021). The eUTXO design allows for simplified development of smart contracts which allows users to create DApps in less time & effort as compared to Ethereum. As of October 2021 however, Cardano has only recently enabled the functionality for use of smart contracts (Sinclair, 2021) which means that there hasn't been a lot of time for it to be tested as a viable means as compared to the like of Ethereum. Due to many failed launches of this same functionality (Klent, 2021) does not instill a lot of confidence that any development time should be poured into Cardano vs the likes of Ethereum. However, Cardano has an advantage over Ethereum regarding scalability since it makes use of a multi layered framework. This framework consists of a different and separate layer for settlements and computational workings (Shrimpy, 2021) The settlement layer as the name suggests, settles transactions as they happen on the network, in the meanwhile the computational layer provides the usage of smart contracts. As of writing the benchmarks for Cardano's scalability has not been implemented and tested quite yet so there isn't a definitive answer on which technology wins out in terms of scaling. For these reasons listen we can discern that Cardano is not well suited to the issues at hand as a lot of it's plans or promises are still yet to be worked upon, Ethereum seems to win out in the fight between the two. If this project was to be revised a look into what changes have been made to Cardano would be of great interest.

3.2.4 Hyperledger Fabric

Hyperledger is an open-sourced development technology that is focused on the development of blockchain solutions. It is hosted by the Linux Foundation, and several leading figures in the technology, banking, supply chain, and manufacturing industries. It was initially first launched in 2016 just one year after the like of Ethereum. Of the seventy-plus groups that the Linux foundation has helped launched it is one of the fastest growing the foundation has been involved with (Hyperledger Foundation, 2021). Hyperledger leverages the blockchain technology for businesses mainly and is designed in such a manner that allows for implementation of services that promise resilience, confidentiality, and scalability (Edureka, 2021). By using Hyperledger, it is possible for a user to make transactions of information that are completely confidentially to other users, as illustrated by Fig. X.

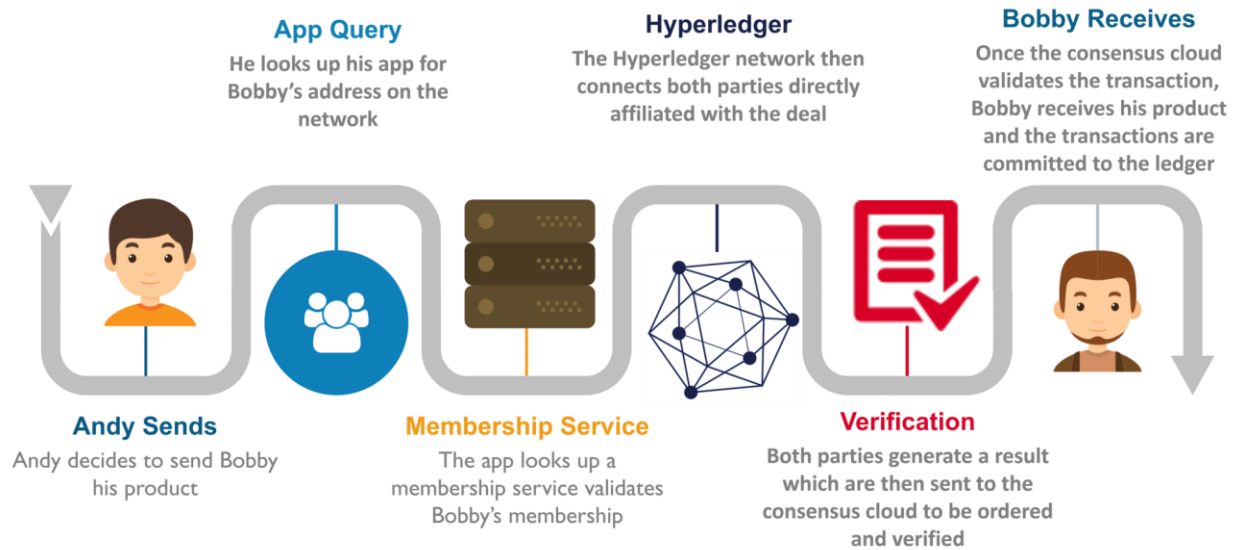


Fig. X (Edureka, 2021)

Hyperledger claims that transactions such as this are not possible in the Ethereum framework, as the technology promotes transparency, allowing all users on a network full access. Hyperledger does not require any currency to operate. This allows for scrabble consensus algorithms to be implemented. It makes use of a term called “chain code” which is essentially its own wording for what is referred to as other blockchain as a smart contract. This chain code handles the same way to a smart contract and is typically coded by making use of a language called Golang, which was created by Google (Google, 2021).

To conclude, is Hyperledger the right choice for the proposed framework? It solves for a lot of scalability, performance, and privacy issues with use of elevated control to access, along with its modular design to customize it to whatever issues are at mind so it could be considered a serious contender for the proposed solution.

3.2.5 Ethereum & Smart Contracts

Ethereum, which is a public based distributed ledger, like bitcoin makes use of two different accounts. These being Externally owned account (EOA) and the other being contract account. Each of these is comprised of an index , which is a 20-byte long address, and a set of private and public keys (Nguyen, 2019). To make use of the Ethereum network a user needs an account first. The currency used by Ethereum is referred to as Ether. Ethereum makes use of smart contracts,

which work similarly to that of a self-operating computer application (Wood, 2014). These operate by making use of contractual preset parameters in their source code. Users can interact with the smart contracts via an application binary interface (ABI)(Nguyen, 2019; Buterin,2014). This code can be executed when a transaction occurs to trigger an event within blockchain to perform a set task. This allows for the development of decentralized applications (DApps). This is beneficial in a healthcare environment as it solves for some issues of interoperability and authentication (Schnitzbauer, 2021). Ethereum provides a decentralized virtual machine to carry out the effects of its smart contracts. Smart contracts themselves were proposed by Nick Szabo who first coined the idea of digital contracts in 1996 (Szabo, 1996). All users are permitted to execute contracts without third party involvement. It also cannot be interrupted from running. Fig. 8 shows the structure of an ethereum transaction.

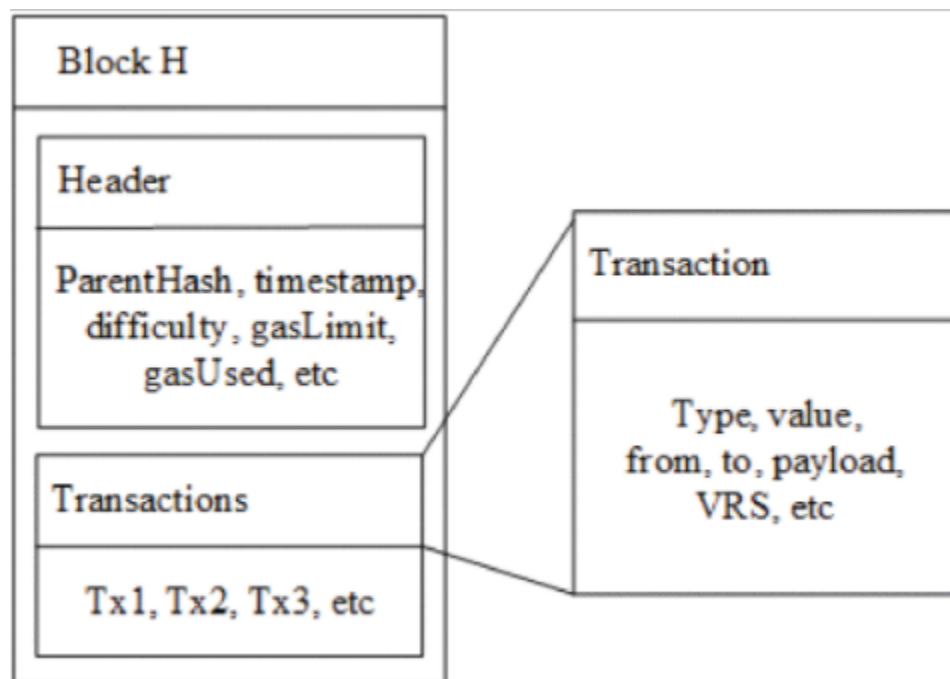


Figure 6 Ethereum Transaction Structure (Lv et al., 2019)

The virtual machine that Ethereum uses is referred to as Ethereum virtual machine (EVM) and is based off Java's JVM (Java Virtual Machine). It is stack oriented, making use of the LIFO (Last in first out) principle. EVM makes use of 'Gas' to operate. Gas is the amount of Ether required for a smart contract to be transacted (Buterin, 2014). Ethereum makes use of 3 different networks.

These include the main net, test net, and private net. The main net is the most used network for ethereum where public transaction occurs. Test net is for developers who need to test their applications, here contracts can be transacted with no concern for paying gas fees. Private net is what it sounds like, this is reserved for private use by parties who prefer not to be on a public network. From a privacy view, ethereum makes use of cryptographic functions to hash as a means of protecting its users (Chen, 2020). This however does not guarantee that these cryptographic mechanisms are 100% secure due to Ethereum's nature as a public blockchain (Apostolaki et al., 2021). As mentioned previously in Fig. 8 a Ethereum transaction is a set of information used to transfer ether from one address to another. The structure of a transaction can be displayed as follows.

{Account Nonce, address of recipient, Gas price, Gas limit, Ether value, sender's signature, data field }.

Below is how a transaction would be formulated:

Transaction Tx = new Transaction(getFromAddress(), gasPrice, gasLimit, to, value, data)
Equation 1

The idea of using smart contracts to carryout functions makes it the main contender for the proposed framework for the prototype to be developed. Since smart contracts can be coded to perform whatever needs to be done, we can design the system around Ethereum, as so it is important to carry out reviews into the technology with this in mind. This means that issues surrounding the usage of Ethereum should be reviewed and any toolkits that can aid the development be surveyed to achieve the design goals that will be set out.

3.3 Taxonomy of blockchain systems

To understand the technology better it is crucial to understand the different classifications of blockchains. Blockchain, can be classed into several different categories, these include: permissionless and permissioned (Iredale, 2019).

3.3.1 Permissionless / Public

In a permissionless blockchain users are all anonymous or have nicknames associated with them. This allows for efficient use when the identity of the user is unimportant to the exchange. Therefore, it is not suited for use in the healthcare sector as the instability of unformatted chains can lead to clogs in the system (Dubovitskaya et al., 2017).

3.3.2 Permissioned

The latter, permissioned, shares a lot of different characteristics to permissionless. Permissioned operates under the principle that the users and nodes involved in the chain must be verified and can be traced back to. This is useful for the healthcare sector in many ways such as legal compliance or privacy concerns (Dubovitskaya et al., 2017). Since the proposed solution aims to keep the names of its users anonymous it is likely that the ideal taxonomy of the prototype would make use of the permission style of blockchain. In this way users could be associated with an ID rather than a name, solving for issues revolving around the doxing of patient's personal information. Fig 9 illustrates the characteristics of both.

Characteristic	Permissioned	Public
Consensus	PoS, PBFT	PoW
Currency	optional	required (incentivization)
Access	by invitation	anybody
Data immutability	medium	high
infrastructure	decentralized	distributed
Throughput	high	low
Privacy	private from non-participants	pseudonymity

Figure 7 Characteristics of Permissioned vs Permissionless Taxonomy (Xu et al., 2017)

3.4 Blockchain Storage Solutions (Off-chain Vs On-chain)

Since blockchain was designed for small scale transactions, the data capacity it can hold is limited. For Bitcoin, the average block can only hold up to one megabyte of data (Jin et al., 2019). Hence it is insufficient to store large records or multimedia file formats such as images. Further, there

are many other aspects that need to be considered when designing a system such as this which are.

- Regulations, such as the General Data Protection Regulation (GDPR) for European users cause issues. A proposal has been made stating that an off-chain solution would comply with the requirements set out by this regulation (Humbeeck, 2017). Within this solution, all data sensitive materials would be stored in a database that is not on the blockchain network, but rather, would be point to in the blockchain. Furthermore, on-chain stored data cannot be altered or deleted because of the nature of the chain being a perpetually growing public ledger.
- Some information has its own 'life cycle', meaning that it makes it unnecessary to store that data permanently. This, is being enforced by several different data protection laws such as GDPR mentioned previously (Esposito et al., 2018)

Fig. 10 illustrates how off-chain storage is implemented. Here two separate applications are using databases and a blockchain network in order to achieve functionality

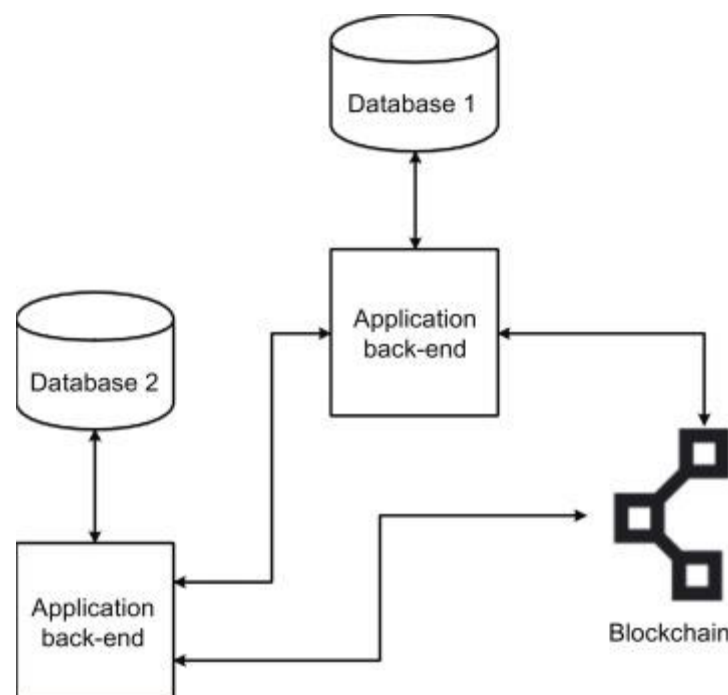


Figure 8 Off-Chain blockchain(Onik et al., 2019)

The advantage of an off-chain solution is that it upholds data integrity. This is done by having the hash of the data on the blockchain itself but storing the data on a separate database. This comes with its own drawbacks however, as privacy issues of the database used are still prevalent. Fig. 11 shows how an off chain blockchain system may work for the healthcare sector.

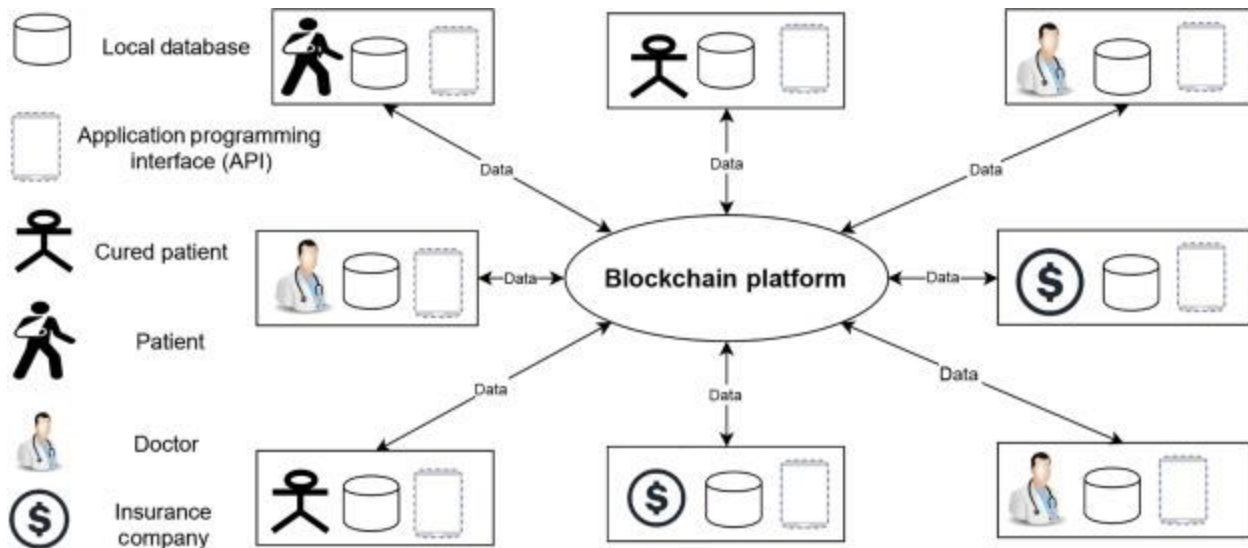


Figure 9 A GDPR compliant blockchain solution (Onik et al., 2019)

Blockchain by itself, is a transparent, secure, and public ledger, that can be used to guarantee the integrity of its on-chain information (its blocks and transactions). Meaning, it can be used to guarantee the security of medical data, if the on-chain design route is taken. Nevertheless, this approach can lead to some poor performance and issues for the system in general. This is since each user would need to themselves download all chain transactions and its blocks locally. For this reason, many researchers have expressed that an off-chain solution is better suited to store medical information (Fan et al., 2018; Xia et al., 2017; Yang & Yang, 2017; Azaria et al., 2016). While queries regarding data and their hashes are stored on the chain for authentication, the data itself should be stored off it. Since the proposed is being used for a national sense, it is important to design a system that is compliant with privacy regulations. It is clear that the off-chain approach should be taken regarding the storage for the application. Since the use of a non-blockchain technology is being implemented to store the data itself it is necessary it is important to secure both points. Use of data encryption within the proposed would most likely solve for any issues surrounding an off-chain solution.

3.5 Data Encryption & Cryptography

Since it seems that storing data off the chain is the approach that should be taken, it is important to properly decide on what cryptography mechanisms should be used to ensure that the data is also secure. This should be done to ensure that the system fulfills its goals of privacy and security. However, should the data that is being stored be encrypted? Surveys have shown that up to 50% of all data breaches belong to the healthcare sector (Mobi Health News, 2018). With 90% of all healthcare organizations having some form of data leak at some stage (Mobi Health News, 2018). It is then apparent that storing medical data would eventually succumb to leaks, mainly due to the following reasons.

- Once a healthcare system becomes endangered by attackers, all data typically is leaked.
- Despite efforts to control access to records, IT staff have inherent access over the data, compromising the overall confidentiality of any system

Because of these reasons, it is obvious that there is a need for encryption within such a system. This encryption could often be the last line of defense for a compromised system. This is due to the malicious attacker not being able to do anything even if they obtained the information without its respective key to unlock its contents. As mentioned previously, permissioned and permissionless blockchains differ regarding the protocols that they employ. This means that, depending on the choice of permission vs permissionless impacts the throughput, mining times, access, and policies involved. But which of these solutions should be used for the proposed system. Throughput is probably the most important metric when it comes to the overall performance of the systems. Blockchains such as Hyperledger Fabric iterate through up to 10000 transactions per seconds (TPS), comparatively Ethereum and Bitcoin have TPS' of 20 & 7 respectively (Maharjan, 2018). The TPS of both Ethereum & bitcoin are too small for the fast-paced nature of the healthcare sector. To counter this, advancements in the technologies are making up for these shortcomings. The Casper version of Ethereum known as Ethereum 2.0 adopts the PoS algorithm and boasts an increase in its TPS throughput to nearly 100,000 (Cryptopedia, 2021). A challenge associated with using permissionless for healthcare information revolves around cryptocurrency. Since for a system to work a crypto incentive is needed, it

causes a few issues for the service. The fast nature at which data is exchanged means that a lot of cryptocurrencies is needed to keep the gears moving. A possible way around this is to create an altcoin (a non-bitcoin token) which can be issued to miners to keep the system working. Since storing of data on chain is off the table due to blockchain low memory capacity per block, an off-chain solution is needed. But this becomes its own challenge when deciding how to secure this information. This section reviews some cryptographic primitives that can be used to aid in access controls.

3.5.1 Broadcast Encryption

Broadcast Encryption (Fiat & Noar, 1994) is a system in which a user can encrypt small pieces of data to a subgroup of people. The users in this subgroup can decrypt the messages sent to them to make sense of the data. In cryptographic storage, keys are encrypted instead of the contents of the data itself (Popa et al., 2011). This key is then decrypted and used to access the information instead.

3.5.2 Identity-Based Encryption

Identity-based encryption (IBE) was first coined in 1984 (Shamir, 1984). It works around the idea that a string can be used as a public key, it was then upgraded by using Weil pairing & elliptic curves (Boneh & Franklin, 2001). Using IBE, a third party is referred to as the private key generator (PKG), this user then generates a 'master' public-private key pairing. In essence, using this a person can then with the public master key, identify the identity of a user by combining it with their key. To get the private key, the user with an identity ID needs to converse with the PKG, who uses their master private key to generate the private key for a given identity.

In this way, IBE gets rid of the need to distribute public keys to its users. It means that any users can communicate securely without the need for key exchanging. This system is ideal for the sharing of data within a private group like an organization.

3.5.3 Attribute-Based Encryption

In many instances there is a need for an information exchange with two parties without prior knowledge of who the receiver of the data will be. This is referred to as attribute-based encryption (ABE). If for example, a patient wishes to share their data with a doctor. This 'doctor'

attribute can be used by the patient to grant access to their data based off this. The data is then encrypted with access only being permitted to users with the attribute 'doctor' (Sahai & Waters, 2005).

ABE is effective when it comes to access control of information. Furthermore, it can be split into two additional categories

1. Key-Policy attribute-based encryption (KP-ABE) is when the keys are linked to the access granting policies and ciphertext is used with an attribute (Goyal et al., 2006)
2. Ciphertext-Policy attribute-based encryption (CP-ABE) is referred to when keys are linked with a set of attributes. Ciphertext is then linked to access policies (Goyal et al., 2006).

Using either of these, a governing authority is present to both issue and validate private keys. This makes them unsuitable for a system in which data exchanges occur across different domains. Additionally, to address the issue of the single governing body. The Multi-Authority attribute-based encryption (MA-ABE) scheme is used (Lewko & Waters, 2011) to ensure no central body is needed, ensuring resistance against collusion.

3.5.4 Search on Encrypted Data

Search on Encrypted Data (SSE) is used to stop keyword searches when it comes to outsourced data that has been encrypted (Poh et al., 2017). The basis of SSE means that a masked index is sent as metadata

3.6 Performance

High compression ratios of data have been shown to affect the overall performance of a system (Lee et al., 2018). The overall design of the system and authorization of data transactions may rely on manual approvals from users which would heavily impact performance (Shen et al., 2019; Mamoshina et al., 2018). The need for upgrades, the computational loads from sensors, disk space, and the overall setup of the network are all noted as issues that affect the scalability of systems (Kaur et al., 2018; Zhang et al., 2016; Zhang et al., 2018). This includes the amount of currency required for the system e.g., Ether for use with smart contracts. It was also found that using globally executed smart contracts can be a factor that sets-off high performance costs

(Dagher et al., 2018). Furthermore, little studies were carried out regarding the management of nodes on the network. These can cause issues when it comes to the number of said nodes on a network, the latency between them, and the number of malicious users (Hyla et al., 2019; Fan et al., 2019). When it comes to testing the prototype the main things that will be evaluated is the overall privacy protection and security it offers but it is crucial that other things are tested to. As mentioned above the overall performance of the system is crucial for the testing to be a success. Furthermore, the scalability of said system is a concern of the technology and should appropriately be tested to ensure the viability of the system. Since there is a huge transfer of data in the healthcare sector at any given time it is imperative that the system does not get bottlenecked, as so tests should be carried out to determine if the technology is suited or not.

4 Methods & System Design

For the prototype, the Ethereum blockchain platform is being implemented (Ethereum, 2018). The advantages it provides surrounding flexibility, as well as its employment of smart contracts makes it the best suited solution for the system proposed.

4.1 Design Goals

To allow for efficiency and security when using the system under the proposed system architecture, the developed prototype should achieve the following design goals.

1. The system should allow for authentication and user identity properties. The prototype should ensure strict verification access to promote system trustworthiness when used. It should also allow only authorized users access to medical records of a given patient, while also preventing malicious threats to the system.
2. The prototype should provide a way to quickly retrieve data in an efficient manner, implementing a level of user authorization by using smart contracts. The users access control design should be designed in such a way that it is lightweight to avoid congesting a network, causing the system to run slower than desired.
3. The system should provide a high standard of security, data protection, and provide flexibility in order to ensure it is feasible to as many healthcare scenarios as possible.

To achieve all of this, the proposed solution makes use of access control scheme such as the use of smart contracts to manage user access. A peer-to-peer cloud storage solution as seen in interplanetary file system (IPFS) allows for decentralized data storage, which is used to allow for elevated access control. These technologies overcome the need for a centralized storage solution, which helps to achieve a level of performance and avoid low system latency. Besides this, along with the coding for the smart contracts, front end system design is carried out by using a mixture of Web3js and react.

4.2 Proposed Architecture

The overall system is designed around the idea of a patient who is in control of their own information. These records may include personal information and medical history of said user. All users within the system have their own personal health ID, this ID is used to identify users on the system without revealing details such as their name, promoting anonymity. Since all users have an ID, it is important to differentiate the different type of stakeholders, e.g., patients, doctors, system admins. All IDs then map to the users Ethereum address. Fig. 12 shows the bases of the proposed architecture. Here we can see that both the patient and the doctor type users can access the same file system as each other.

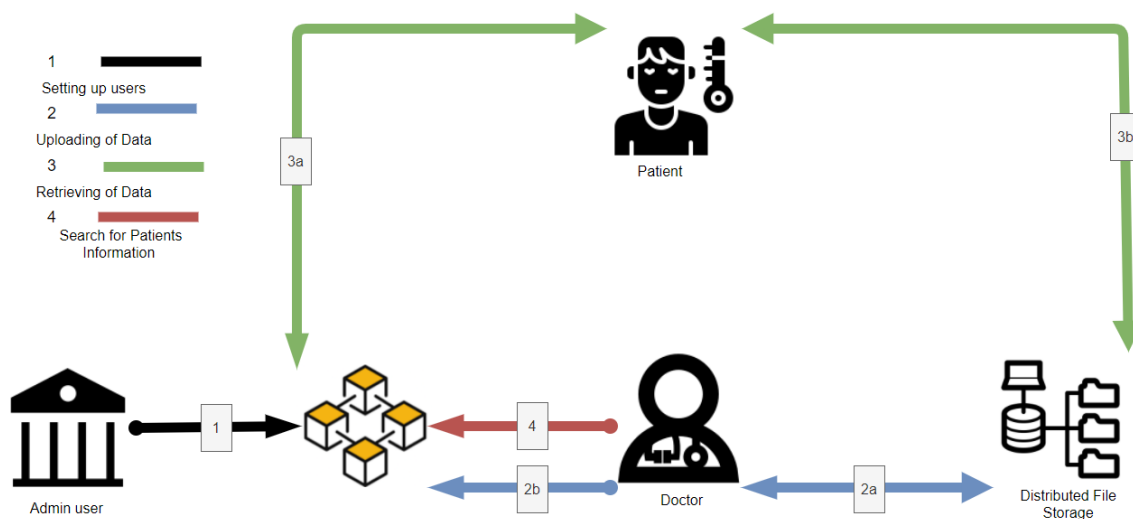


Figure 10 Proposed architecture

4.3 Different Stakeholders

This section outlines the different stakeholders or user types that are used within the system.

4.3.1 Admin users

A national body such as the National Health service would need to set up admins users to play a critical role in allowing for patients to be given a unique ID to be able to use the proposed blockchain solution.

4.3.2 Doctors

Any Doctors would play a role as both a viewer and as an issuer of data. Doctors would play a key role in issuing patients with their records when coming to them for health-related issues. The role of a viewer would occur when a patient comes for tests etc.

4.3.3 Patients

Patients oversee their own credentials and data for the system. Patients must provide access to records and a unique ID when interacting with any doctor. This extends to visitation of any new doctor they have not previously visited as they would not have access to their data. In this scenario the patient acts as a viewer of their own data but also provides access to who they want.

4.4 Technologies used

This section gives an insight into the different technologies that went into the development of the prototype. There are a multitude of different ways to design and implement a blockchain, given that the technology is yet to fully flourish. The approach used in the proposed was made using the following.

4.4.1 Ethereum

The proposed Blockchain solution makes use of Ethereum as previously mentioned. The flexibility and adaptability it offers over the likes of other blockchain technologies such as bitcoin or Hyperledger, makes it the perfect use for the development of a healthcare-based application (Ethereum, 2018; Wood, 2021). Since Ethereum makes use of blocks and smart contracts we can use these to an advantage to program a system around the protection of patient's data. Furthermore, Ethereum uses the PoW algorithm that was discussed previously. Within this

algorithm, miners are employed to help provide validity, thus making it the more ‘tamper-proof’ of the different algorithms used by blockchain. The role of Ethereum in the solution is to ensure every user in the system is given a unique Ethereum address, this address is a hashed version of their public key, since Ethereum associates a pair of keys for each account. To interact with the prototype users must access the Ethereum network via their private keys and without this key are unable to access the system, meaning that malicious users cannot pretend to be a user they are not to probe for data. Anytime a transaction is carried out, a record of said user is first validated and then stored within the blockchain. Since the smart contracts require inheritance from other contracts it leaves little room for exploitation, since the network is set-up in a careful way to leave no room for exploitation from malicious intent.

4.4.2 Web3.js

Web3.js provides an extensive collection of different libraries that can be used and implemented to interact in the front end with an Ethereum blockchain (Web3.js, 2021). It also provides support with meta mask, which will be discussed. This support allows for a connection between the two by making use of a ‘Remote Procedure Call’ protocol, to allow it to connect to the blockchain application. In this way Web3.js allows a user to carry out a lot of tasks such as sending Ether from one address to another, deploying smart contracts, reading, or writing data from smart contracts, and to call function within different deployed smart contracts (Web3.js, 2021).

4.4.3 Next.js & React

Next.js (Next.js, 2021) is a React (React, 2021) framework that is used to help build a configured web-based application. This approach allows for functionalities such as modular programming, and routers built in allowing for routing within these applications. Use of a system called ‘Hot Module Reload’ means that any changes made to the application are based in real time without the need for restarting the overall service (Next.js, 2021; React, 2021).

4.4.4 Meta Mask

Meta Mask is an extension for browsers that acts as a wallet and a way to interact with the Ethereum blockchain. It allows the user to sign in and sign transactions from a unique address (Meta mask, 2021).

4.4.5 IPFS

IPFS is a peer-to-peer file distributed system used mainly for storing data. When data is uploaded to it, it returns a hashed version of the data. This is used for the off-chain solution (Labs P, 2021). It is also built on top of torrent technologies such as BitTorrent and is being used within the prototype as an alternative to a centralized server (Nguyen et al., 2019). Within the prototype users make use of IPFS to store data. This allows for high throughput regarding storage as well as data retrieval (Confais et al., 2017).

4.4.6 Solidity

Solidity is an Object-oriented programming language which is used to write smart contracts for use with various blockchain platforms, most notably Ethereum. The language compiles programs to then be used to run via the EVM (Solidity, 2021).

4.5 Implementation

There are three different types of stakeholders within the system, these include a patient user, doctor, and an administrative user. Each user of the system has a unique ID (ID_u) which is allocated to them by a national health care body via the admin user. Users make use of an ethereum account address which is subsequently used as their ID_u . This ID_u is mapped within the frame so that each user has an associated attribute assigned to them depending on what stakeholder they are. Within the developed framework different dashboards are available to different stakeholders to carry out specific tasks. Since ethereum provides a unique ID in the way of its address, these are used in place of the ID that would be issued by the admin user. The ID can then be mapped in such a way that it is able to identify different parties from each other. Within the prototype each different party has their own user interface they can use, which is given a set of functionalities suited to the party. Medical records are private data owned by the patients and when it comes to viewing or sharing of such information, privacy is of the utmost importance. The proposed architecture makes use of an off chain distributed file storage solution, which offers two main advantages. The first of these is the offloading of the blockchain technology, which allows for improved scalability. Second, this kind of solution cuts the costs of the framework overall. However, there are some downsides to using this method. Mainly in the

fact that the off-chain solution could then become the main point of attack for malicious users, potentially causing compromised privacy (Zhao et al., 2020). By bolstering the architecture with privacy protection techniques attackers should not be able to access any information, nor gain any insights that may lead to the identification of a patient or any other users. Therefore, the goal is to enhance the privacy of the overall framework by securing the off-chain solution in order to ensure that the patients have full control over their data. This architecture makes use of 4 main functionalities in order to carry out its job. These are setting up of users, uploading of data, retrieval of data, and lookup of patient's information. All interactions that take place between the different stakeholders are subsequently recorded on the blockchain as a form of an immutable transaction.

4.6 Functions

To achieve the prototype, there are several different functions needed to carry out the work, which are stored within the different smart contracts. These are as follows. To understand some of the methods used within the system it is important to first refer to Appendix [1](#), [2](#), & [3](#).

4.6.1 Registering Patients on the system

This functionality is provided to enable administrative users the ability to set different users up on the system. Each user is provided with a unique ID (*IDu*) which in turn is provided to them when they sign up to use the proposed system. Once this ID is assigned, the user will have this identifier for their whole life. This ID is then digitally signed and transacted onto the blockchain. The exact way that this ID is setup and distributed is beyond the scope of this work as it is the duty of the national bodies who looks over the healthcare sectors jobs to formulate an appropriate policy to do so.

4.6.2 Signing up & Login of Users

After registering for the system, the user should in turn be informed to set up their own Ethereum account with its own unique address for use with the system. This account is then linked to their *IDu*. This leads to the need to set up a private and public key pairing to ensure privacy needs. By making use of ECQV and certificates, a key pair can be set up from an administrative user in charge of setting up records. This has an advantage that only the user themselves is aware of

their own private key and the relation between IDu and their public key can be made. Further to this, by using ECQV users can find the public key of different users given that IDx (x refereeing to the user) and $certx$ are known.

4.6.3 Uploading data

The submission of data is split up into different steps. The first of these involved is when a patient receives a new record $data_i$ is assigned to them. When this happens, the assigning doctor computes the hash of the data $OH = H(data_i)$. This hash is referred to as the 'Original hash' and is used to promote integrity within the system. Secondly, the doctor encrypts the data using the public key of the corresponding patient they are seeing by performing $data_e = E_{Q_{patient}}(data)$. This encrypted data ($data_{i_e}$) is then sent to the distributed file storage. This results in an *address* being returned. Finally, the doctor creates:

$$M = (address, OH, metadata, Q_{patient}) \quad (2)$$

They then sign the message M and transacts the result $\{M\}_{d_{doctor}}$ to the blockchain.

4.6.4 Viewing and receiving data

To then view information on the blockchain, the patient can send a request off and gets an *address* back from the distributed file storage. The data is received in an encrypted format, using their private key they can decrypt and view the data by carrying out the following:

$$data_i = D_{d_{patient}}(data_{i_e}) \quad (3)$$

4.6.5 Giving access to data

No one should be able to access the data of the patient unless they explicitly give access to it by themselves or an appointed guardian. To do so patient carry out the following steps. The patient calculates the hashed values of their data by performing

$$H_{TP} = H(data_i) \quad (4)$$

this is then encrypted by using the public key of the intended recipient by carrying out

$$data_i^{TP} = E_{Q_u}(data_i) \quad (5)$$

Then, the patient uploads $data_i^{TP}$ to the distributed file system and is returned with an *address*. Usually, a patient would not have access to add to their medical data, but in this instance to ensure the protected privacy, the patient may be allowed to submit a ‘*temporary record*’ of an already issued record. The patient then creates a message by carrying out the following

$$M = (address, H_{TP}, Q_u, metadata) \quad (6)$$

This is then signed, and this results in

$$\{M\}_{d_{patient}} \quad (7)$$

Which is subsequently transacted onto the blockchain. Since it is totally possible for another malicious user to try and use this to submit false information the system carries out the following comparison:

$$H_{TP} = OH \quad (8)$$

Since OH is the hash value of the same data computed by a doctor, if the two are the same it means that integrity has been upheld. If it matches the system carries on, if not the temporary data is removed. Then, by using the *address* the doctor who wishes to be given access downloads $data_i^{TP}$ from the distributed file storage. This download is then decrypted using their private key by doing the following

$$datai^{TP} = D_{patient}(datai_e^{TP}) \quad (9)$$

The temporary data upload is then discarded as it is not needed further to this.

4.7 Front-End Features

Users of the prototype interact with it by the use of a web browser that requires the installation of meta mask beforehand to work properly. This extension allows for a user to connect with applications on the Ethereum blockchain. The user interface is built on top of 'Next.js'. The prototype sends various types of requests that are parsed by the back end blockchain code using functions formulated by using 'Web3.js'. Different front end graphical interfaces users can interact with are as follows. For added protection an external library *ecies* (*ecies*, 2021) is used to assist in the generation of the keys used for encryption and decryption. As well as this another library called *Crypto* which is inbuilt into Node.js is used for generation hashes, ciphers, and verification of messages.

4.7.1 Administrative Dashboard

Admin users are able to set up new patients within the system by entering their Ethereum address along with their identifying details. For these users who are set up on the system, use of a dashboard depending on their attributes are given i.e., doctor or patient. Before uploading data to the blockchain, the administrator or doctors compute for a '*original hash*' *OH*. This hash allows the system to detect for fraudulent users. It is then encrypted with the user's public key before being uploaded to IPFS. IPFS returns a hash which is pushed with the users ID onto the blockchain for future retrieval.

4.7.2 Doctor Dashboard

The doctor's dashboard allows for them to view the records of a patient whose ID they enter. If the ID of the entered patients equates to a user who hasn't allowed them access, they will be given an error message that is recorded by the system. If the doctor is authorized, then they will be allowed to check the records of that given patient. Furthermore, they are able to add a new additional record to a patient's record. The use of smart contracts means that unauthorized access is cutout and logging is improved by giving a scope of who is viewing a given patients records.

4.7.3 Patient Dashboard

The patient's dashboard allows them to view their medical records uploaded so far by different healthcare professionals. To do this, a patient retrieves the IPFS hash values of their information by use of a mapping uploaded on the Ethereum blockchain. This hashed value is downloaded by the patient and is subsequently decrypted by making use of their private key. Furthermore, a patient can make use of an 'allow access' function that allows them to add a doctor based on their ID as an authorized user to access their records and add new records. This function is carried out by use of a modifier which checks a hashed list of address that are allowed to view a given record.

4.8 Back-End using Smart Contracts

All requests being handled from the front end are done so by the backend via the smart contract using ethereum. These requests are referred to as functions and are housed within various smart contracts for use. When a call is made to receive information, IPFS is implemented to receive the needed information. This back end is achieved by making use of the smart contracts as follows.

4.8.1 File contract

Fig. 13 shows the set up involved within the file contract. This contract is used when the metadata of a record is uploaded or retrieved via the Ethereum blockchain. The details of such are the title, description, and the hashed values that are generated using IPFS. The contract has two main functions. The first is to upload the details of a record as a file to IPFS. The second is to download the metadata of files under the given Ethereum address.

File Contract		
Variables		
Type	Name	Description
struct	record	To store the IPFS hash
mapping	userRecords	Contains a mapping of the IPFS records to the patients address
Functions		

Name	Description
uploadFile	Sends data to be uploaded to IPFS
getUserFromAdress	Receives the data back from IPFS

Table 1 Variables & functions of the File Contract

4.8.2 Admin User Contract

This contract is used to create and store / retrieve the details of users using the blockchain. Fig. 14 shows the different variables used. There are also different functionalities and provide some useful functions. The first of these is a function that checks if a user is already on the network. Secondly If the user is not on the system, create their credentials within the blockchain. Finally, If the user exists the users can retrieve data from the blockchain.

Admin User Contract		
Variables		
Type	Name	Description
Struct	user	Stores the names and attributes of the users
mapping	userFromAddress	Maps the address to the user
mapping	addressFromUser	An array of users' Ethereum Address
Functions		
Name	Description	
createUser	If the users isn't already on the system, then their details are stored within the blockchain. This also adds their address to a mapping and vice versa for their ID.	
getUserFromAddress	Gets the users details given the address	
getAddressFromUser	Gets the Address given the user ID	
checkUserExists	Checks if the user ID already exists	
checkAddress	Checks if the address is already on the system	

Table 2 Variables & functions of the Admin User Contract

4.8.3 Doctor Contract

Fig. 15 shows the overall set up of the doctor contract. This contract allows for the uploading of a file or record to the IPFS file storage and additionally is used to retrieve data. This data retrieval

can only be carried out by a user who has been authorized to do so and failure to gain said access can lead to an error message being returned and recorded. To do this the doctor contract uses an instance of both the patient and file contract to gain the core functionalities needed to carry out its job.

Doctor Contract (Inherits the Admin User & Patient Contracts)		
Variables		
Type	Name	Description
address[]	patients	A public array that houses all patient addresses
fileContract	fileContract	An instance of the deployed file contract
patientContract	patientContract	An instance of the deployed patient contract
Functions		
Name	Description	
doctorContract	Constructor used to create instances of the user & patient contracts using the address of the other deployed contracts	
getRecord	Get record for a given patient, this will only work if the user has been authorized by the relevant patient	
addRecord	Adds a new record for a given patient, this will only work if the user has been authorized by the relevant patient	

Table 3 Variables & functions of the Doctor Contract

4.8.4 Patient Contract

Fig. 16 shows the setup of the patient contract. This contract is used to obtain the data of a patient. This contract also makes use of an instance of the file contract to carry out functionalities. This contract also allows for the main functionalities of retrieving and viewing of data, that is used by the doctor contract.

Patient Contract (Inherits the Admin User Contract)
--

Variables		
Type	Name	Description
fileContract	fileContract	An instance of the deployed file contract
Functions		
Name	Description	
patientContract	Constructor used to create instances of the admin user contract using the address of the deployed admin user contract	
getRecord	Returns the data of a record or 'file' of the user	
allowAccess	Allows the users to assign a doctor user access to their records by entering that relevant doctors ID	

Table 4 Variables & functions of the Patient Contract

4.8.5 Certificate Contract

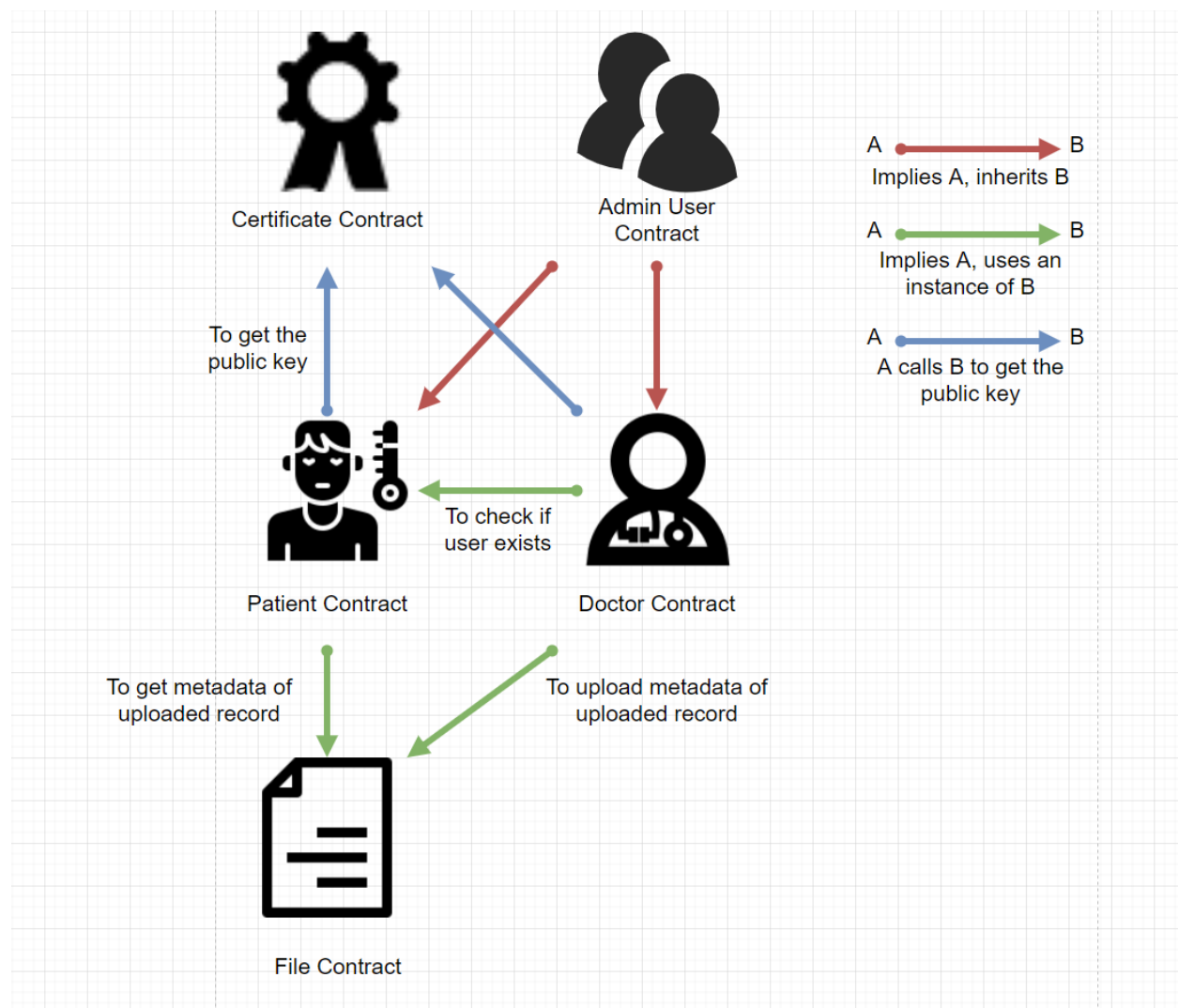
When a new user is setup up to use the system, they undergo the registration carried out by the admin user. When this step is carried out this contract is then used to store the users generated public key on the blockchain. Furthermore, to ensure privacy, this contract is then used additionally to retrieve the public key of a receiver and perform the necessary encryption. The setup of this contract can be seen in fig. 17. This contract is used to implement the uploading of data functionality as well as the registering of a user.

Certificate Contract		
Variables		
Type	Name	Description
mapping	publicKeys	A mapping that houses the public keys of all users, assigned to their address
Functions		
Name	Description	
setKey	Stores a user's public key in the key's array	
getKey	Returns the public key for the assigned address	

Table 5 Variable & functions of the Certificate Contract

4.8.6 Interaction across various smart contracts

To implement the system interaction between different contracts are invoked depending on the user and what they wish to perform. The admin user contract houses different information such as the name of a user and what attribute they hold i.e., doctor, patient, admin, and their ethereum address. This contract, which is inherited by both patient and doctor contracts to allow for some core functionality. The doctor and patient then make use of the file contract. The patient can use this contract in order to view their medical records. Whereas the doctor makes use of the file contract in order to upload new medical records for a given patient. The certificate contract can be called by both the doctor and patient in order to discover the public key, based on a user's Ethereum address. Fig 18 shows the interaction between the different contracts.



5 Testing & Performance

The performance and the viability of using a framework such as the proposed is evaluated by performing different tests using the setup as seen in fig. 19. To carry out said evaluation a test network is used, in this case the Rinkeby testing network is being used (Rinkeby, 2021). Rinkeby is a test network that is provided by the Ethereum foundation themselves. The Infura (Infura, 2021) API is then used to bridge the gap between the application and the network.

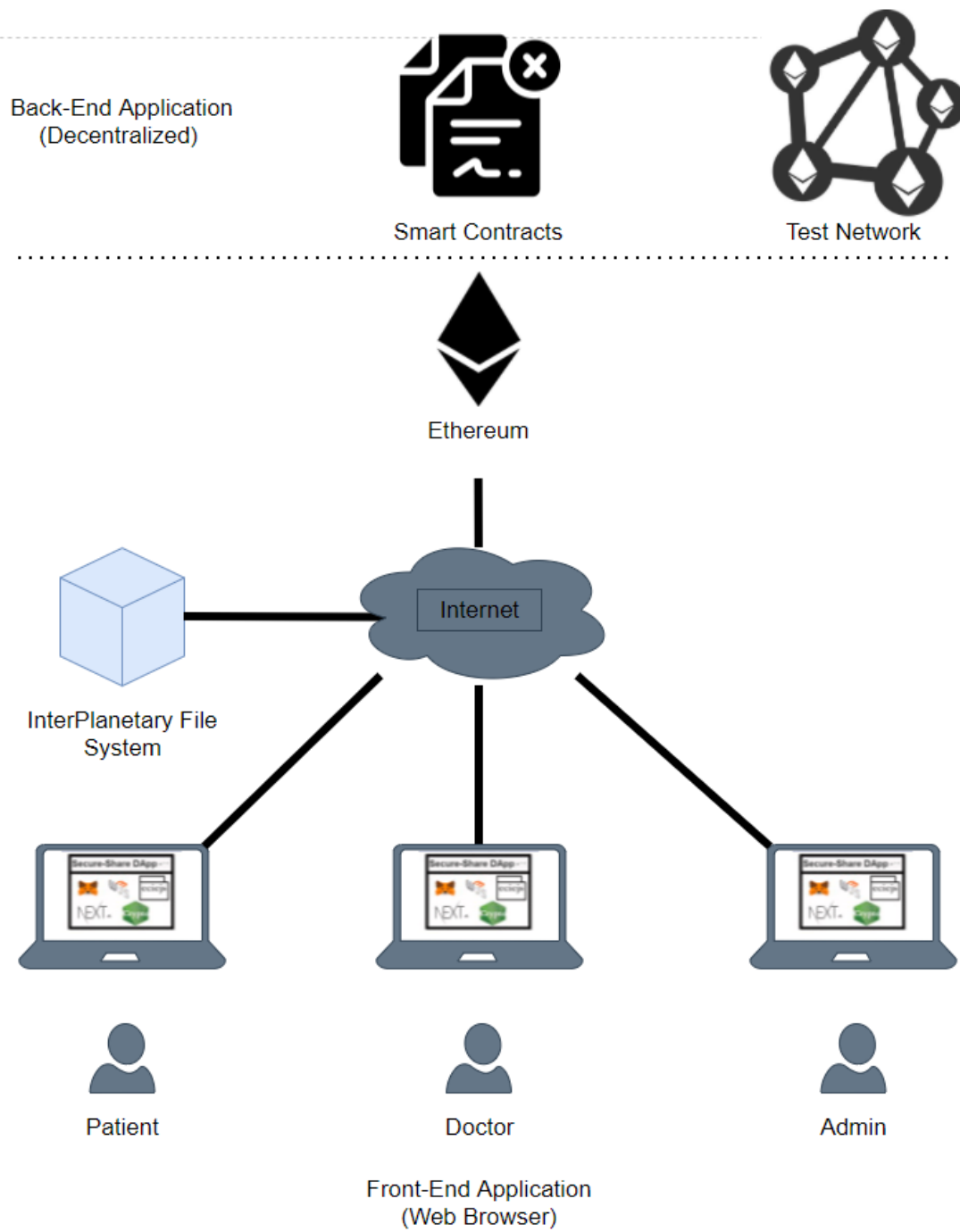


Figure 12 Set-up for desired testing

5.1 The Testbed & Test Data

The data that is needed to work with the example proves for some ethical issues and concerns regarding what should be used. If we use real data, there is the concern that we work against the whole point of the project itself. For this reason, use of the JavaScript library 'Casual' is used

<https://www.npmjs.com/package/casual>

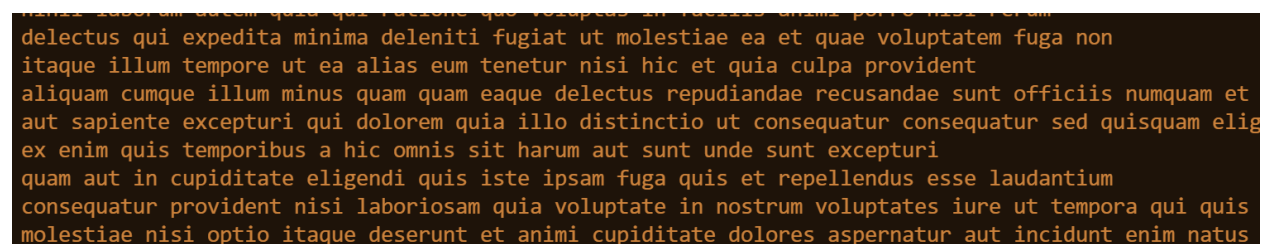
. This package allows for the generation of rather data such as names and random string of words thrown together. Fig. X shows ten names generated using this methods, the details of this code can be seen at appedneix



```
Earlene Stamm
Jefferey Lind
Abelardo Carter
Delpha Satterfield
Velda Erdman
Marc Kuhn
Juliet Gerlach
Jan Bahringer
Lauren Koelpin
```

Fig. X the random generation of names

For our testbed, 100 of these names are generated to use as users for use in our testing process. As mentioned, previous, random strings are also generated as shown in Fig. X



```
delectus qui expedita minima deleniti fugiat ut molestiae ea et quae voluptatem fuga non
itaque illum tempore ut ea alias eum tenetur nisi hic et quia culpa provident
aliquam cumque illum minus quam quam eaque delectus repudiandae recusandae sunt officiis numquam et
aut sapiente excepturi qui dolorem quia illo distinctio ut consequatur consequatur sed quisquam elig
ex enim quis temporibus a hic omnis sit harum aut sunt unde sunt excepturi
quam aut in cupiditate eligendi quis iste ipsam fuga quis et repellendus esse laudantium
consequatur provident nisi laboriosam quia voluptate in nostrum voluptates iure ut tempora qui quis
molestiae nisi optio itaque deserunt et animi cupiditate dolores aspernatur aut incidunt enim natus
```

Fig. X shows the random data

This can also be seen within the code at appendix X. Since this data is totally random, we don't run the risks of accidentally leaking any person's information. Once this information is obtained it is recorded in a CSV file format to keep track of. Each of our created users is then assigned 10 different 14 string long pieces of data to act as their confidential records. Since the data is recorded in this manner it is easy for testing to be carried out regarding the issues at hand. This data is then uploaded to the blockchain test network. This process takes upwards of two hours to upload around 1000 different records to the prototype. This uploading process is also documented and used to evaluate the asynchronous properties of the design; this is explained further later in section 5.5.

5.2 Privacy & Security Analysis

The Open Web Application Security Project (OWASP) outlines the most found vulnerabilities when it comes to the security involved in an application such as this (OWASP, 2017). Almost all the issues addressed are also found to be prevalent in the blockchain architecture as well (Poston, 2020). This section will compare the performance of the prototype to the issues outline, and critically discuss if the model can overcome them.

5.2.1 Injection

Refers to when an attacker injects their own scripts and malicious content into a system to have them execute without need for authentication. Examples of such are Structured Query Language injection, Operating system command line injection, and server side include injection (Seng, 2018). Usually, systems require in depth testing to counteract these measures. Since the Dapp prototype is setup in such a way that only authenticated users can access it there is no way to perform these actions, even if a user with proper authentication was able to do so, a trace of their address and credentials would lead back to them.

5.2.2 Broken Authentication & session management

Lack of user authentication of different sessions leads to attacks on users, allowing malicious users to find out information such as passwords or keys. This would allow for impersonation attacks to occur (Hassan et al., 2018). Whenever a transaction is carried out, Meta Mask verifies the user. Meta Mask itself uses blockchain to store its user credentials which means that unless a user explicitly shares their data, it cannot be obtained.

5.2.3 Sensitive Data leaks

Leaking of data refers to the mishandling of data in an improper manner, this leads to information being revealed that a user might not want it to be. Within the prototype all transaction is encrypted. This makes it impossible for data to be leaked.

5.2.4 Broken Access Control

Improper control of access typically leads to users being able to access records or functions that they would normally not have access to (Hassan et al., 2018). This can lead to malicious usage. Generally, systems would make use of Multi factor authentication to overcome these issues.

However, since users have their unique address, the system is set up in such a way that the user authenticates themselves. The nature of Ethereum smart contracts also lends the ability to test themselves by using the Remix integrated development environment (IDE). By making use of Meta Mask session management abilities on the users end also makes the system more resilient to these issues.

5.2.5 XML External entity

Poor usage of Extensible Markup Language (XML) can lead to malicious intrusions. Proper in-depth testing is required around the security of systems that use external XML entities such as Static application security testing (SAST)(P. A. I, 2015). By making use of Solidity and proper formatting for external entities by using JavaScript Object Notation (JSON) instead leads to a more robust system.

5.2.6 Security Misconfiguration

Issues arise when controls surrounding security is not handled correctly. By making use of Dynamic application security testing (DAST) system are generally combed for any misconfigurations (P. A. I, 2015). However, for the prototype all testing is carried out by the solidity Remix IDE itself. Furthermore, Meta Mask is used on the client end to help enhance the security.

5.2.7 Cross-Site scripting

By using client-side malicious scripts, a user can inject and hijack systems into redirecting them to fraudulent sites. These are combated by using strong validation and encryption of information. Since the prototype ensures proper validation of data by using state of the art code practices there is no concerns.

5.2.8 Insecure deserialization

This refers to the execution of malicious code by a user. This code targets and exposes privileges of objects and aims to corrupt them or gain knowledge on how to attack the system. With the prototype, even if a user were to get logged in, they would not have the necessary authentication to do anything, they would also leave a trace due to the nature of the technology.

5.2.9 Lack of monitoring and logging

Insufficient monitoring of system logs can lead to an eventual breach of security since there is no way to watch how the system is being used. The prototypes blockchain nature means that all transactions of information are automatically logged on the network.

5.2.10 Using components with known Vulnerabilities

Users knowingly targeting a weak part of the system is a concern. By developing systems using a third-party software, systems can become vulnerable as the technology is out of their hands. Generally, in depth analysis is used to understand what issues maybe arise from using third party modules (Huang et al., 2004). Since Ethereum only makes use of trusted platforms such as Next Js and Web3js there is no concerns surrounding this.

5.3 Smart contract-based privacy concerns

Smart contracts may be used for fraudulent activities, this can affect the performance of the overall system (Hu et al., 2021). To ensure that the smart contracts are developed correctly, several attacking methods were researched to counteract their effects (Atzeiet, 2016; Praitheeshan et al., 2019; Sayeed et al., 2020). Examples of such attackers are as follows.

5.3.1 DOA attacks

Decentralized autonomous attack (DOA) refers to a malicious user, who uses a separate smart contract to perform their attack. They call a function with vulnerabilities with their own deployed addresses (Sayeed et al., 2020). Since the prototype does not actually transact any Ethereum itself since (as it is not used for commercial trading) there is no way for this attack to be carried out. Furthermore, an inclusion of a 'gas limit' on the front end of the application also aids in its defense against the DOA attack.

5.3.2 Overflow & Underflow

Numbers are stored as an 'integer' type within smart contracts. If the integer computes as a number lower than the application can comprehend the resulting figure becomes $2^{256} - 1$. Overflow can occur when the opposite of this happens, however the number in this case is converted to 0 instead. This leads to a point of attack for any smart contract-based system

(Praitheeshan et al., 2019). Since the prototype does not revolve around numbers, there is no need for concern about this.

5.3.3 Parity multi-sig wallet attacks

This refers to a publicly available library which functions around smart contracts that create crypto wallets. The malicious user uses this to try and find a way to initialize the library and so gain rights over them (Praitheeshan et al., 2019). Since the system is designed around only being deployed once, contracts cannot be re-deployed, as so there is no concern over this attack.

5.3.4 Rubixi Attack

If a developer fails to change the name of the constructor within the code for the contract it can lead to said constructor becoming a public function. This means that any user would then be allowed to access the privacy controls, allowing them to gain ownership of that contract (Chen et al., 2019). All variable names on the prototype have been checked, furthermore, ownership modifiers have been assigned to make sure this type of attacker can never happen.

5.3.5 Short Address Attack

To buffer addresses, the EVM adds zeroes to the end of its addresses to allow for consistent 256-bit data types. This leads to vulnerabilities surrounding attackers knowingly leaving out numbers. This issue was fixed by the Ethereum team themselves for contracts using version 0.5.0 or newer. Since the prototype makes use of the newest version of Solidity there is no concern from this attack.

5.4 Computation of Costs

In order to safely manage and secure the availability of the core decentralized peer-to-peer networks, the blockchain itself requires incentives to give to miners to carry out its core functions. As so, miners are given incentives for validating the transactions sent to the chain as well as the mining of new blocks for Ethereum. These incentives are given to the miners by the sender of the transaction after they validate. This is referred to as 'Gas'. Therefore, to secure the economic stability of the prototype it is imperative that cost computations are carried out regarding the proposed solution. Two different types of costs are incurred by a smart contract using the Ethereum blockchain which are.

- Transaction Cost: Refers to the gas that is used when a contract is sent along for validation with some data
- Execution Cost: Occurs when gas is consumed to execute a contract. This value depends on the complexity and number of different variables used by the contract

The solidity remix IDE (Remix, 2021) can be used in this case to calculate the costs of the prototype. Table 1 shows the costs involved in the deployment of the different contracts. Additional costs are incurred by the file contract is due to its natures of storing extras hash values on the blockchain, which are the original hash values computed by the doctor over a record at the time of uploading. Since the patient and doctor contract both invoked an instance of file their costs are affected by the previously mention additional costs also.

Smart Contract	Transaction Cost	Gas Limit	Total Cost ($Ether_i$)	Total Cost ($Euro_{ii}$)
User Contract	363557	3000000	0.0003636	€0.94
Patient Contract	903230	4000000	0.0009032	€2.34
Doctor Contract	863230	4000000	0.0008632	€2.24
Certificate Contract	165481	3000000	0.0001655	€0.43
Admin User Contract	476831	3000000	0.0004768	€1.23

- 1 Ether = 10^9 gwei
- 1 Ether = €2,231.59 as of 22/08/2021
- Varies on the metadata length (20 is used for testing)

Table 6 Deployment costs (Gas) involved with the smart contracts

Every smart contract has a 'gas limit', this limit is the max amount of gas that can be used for a specified contract. Different to this, a 'gas price' refers to price of gas represented in the Ethereum currency Ether. The more gas that is used, the higher priority to the miners on the

network, which results in a faster execution time. However, since the sharing of a patients' data is not a time-sensitive scenario, the gas price used is 1 GWEI for all the tests being carried out. This helps reduce the operations costing of the application allowing it to be economically viable.

Table 2 shows the costs being calculated by remix for some chosen functionalities of the application. The main difference between Table 2, and Table 1, is that the latter is used to show the gas being used to deploy the smart contracts on the blockchain, this being a one-time cost as it can be used repeatedly once deployed. Whereas the former symbolizes the transaction costs involved in using some of the core functionalities of the smart contract that have been deployed and transacted to the chain.

Function	Transaction Cost	Total Cost ($Ether_i$)	Total Cost ($Euro_{ii}$)
Registering a user	22432	0.0002243	€0.58
Uploading a $record_{iii}$	334392	0.0004144	€1.07

- i. 1 Ether = 10^9 gwei
- ii. 1 Ether = €2,231.59 as of 22/08/2021
- iii. Varies on the metadata length (20 is used for testing)

Table 7 The cost of the core functions within the contracts

5.5 Functionality

Firstly, by making use of a function data is encrypted. Fig. X shows the return that comes from this, a uint8array of data. This data is signed initially by a public key to secure the data contained.

```

3 C:\code> node index.js
<Buffer 04 d5 01 d7 5d 3b c2 7a 58 69 1e 8f 66 99 20 44 51 58 fe df 8e 3e a2 04 18 ca 98 89 2a 96 77 7b e5 e0 51 2f 25 7
4 50 76 d1 8b f4 70 5a fc 10 dc 9b 6d ... 59 more bytes>

```

Fig. X

To be able to decrypt the data two separate things are needed, the first of these is the buffer seen previously in Fig. X. The other key to this process is the private key paired to the public key used to encrypt the information. Another function is passed these two bits of information and

the data is returned. Within our project blockchain is proposed to solve for issues revolving around privacy, as so it's important to use this in our framework to determine its effectiveness. When the process of encryption of is carried out a toolkit for JavaScript utilizing IPFS is used. This toolkit forwards the buffer to a chosen IPFS pathing, and a hash is returned and stored within the blockchain as a reference to the data. There is no way to determine what is in this hash location thus proving the theory that blockchain enhances privacy. Since the key pairing is stored within the users MetaMask address there is no way for any other users to access confidential records without permission. To understand more about the functionality of the application a look at Appendix 10.4 is recommended.

5.5 Execution Times

The total time to execute a request is made up of different things such as the block mining times, communication time, and lastly the time it takes to execute the smart contract. Since the prototype is built for a healthcare-based system, it is imperative that it is able to handle a large number of requests, for this reason it is important to gauge how well the system performs in this regard. A given user can execute two kinds of requests to the blockchain. The first, is the reading of data from the chain. The second is writing information to it. Reading doesn't involve creating any new blocks; therefore, it takes a small amount of time to execute. However, the writing of information to and transacting it to the blockchain requires miners to then validate the block, which may take some time. According to the Rinkeby test network official website, the average time it takes for a request or transaction to get processed is 15.04 seconds(Rinkeby, 2021). Different types of tests are carried out to evaluate the prototype.

5.5.1 Scalability & Execution of the Prototype

For a healthcare-based record sharing system such as the proposed that deals with a vast number of users, as well as records themselves, the scalability of said system is a very crucial requirement. To determine how the application scales with a growing number of users requests some testing is carried out. To do so the uploading of medical records is carried asynchronously. A thousand different requests were sent out at the same time from the application to the Ethereum network as to gain an insight into the normal working conditions that the proposed may experience. The Rinkeby test network is able to process around 20 different transactions per second, this along

with the average processing time is 15 seconds. Meaning, that all the transactions the network receives (including ones from different applications) get parsed in batches of the max possible transition that can be accommodated. This means that there may be a queue of requests waiting to be processed which in turns means that the values recorded for the performance of the prototype may be skewed. It is worth noting that the parsing times are multiples of 15 since the average time is 15 seconds. This artificial increase to the time to carry out requests could potentially be normalized if a different consensus algorithm was employed as the one currently used to be PoW. This could be obtained by use of a different platform which may be better suited to high computing capabilities that support more time effective algorithms such as Hyperledger as discussed previously (Xu et al., 2021). Fig 21 shows the results of said testing, surprisingly there was no increase or decrease in the amount of time that it took to upload records. This information leads us to believe that there is a good deal of promise when it comes to expanding the prototype further since the 1000 iterations caused no concerns regarding its ability to perform asynchronous.

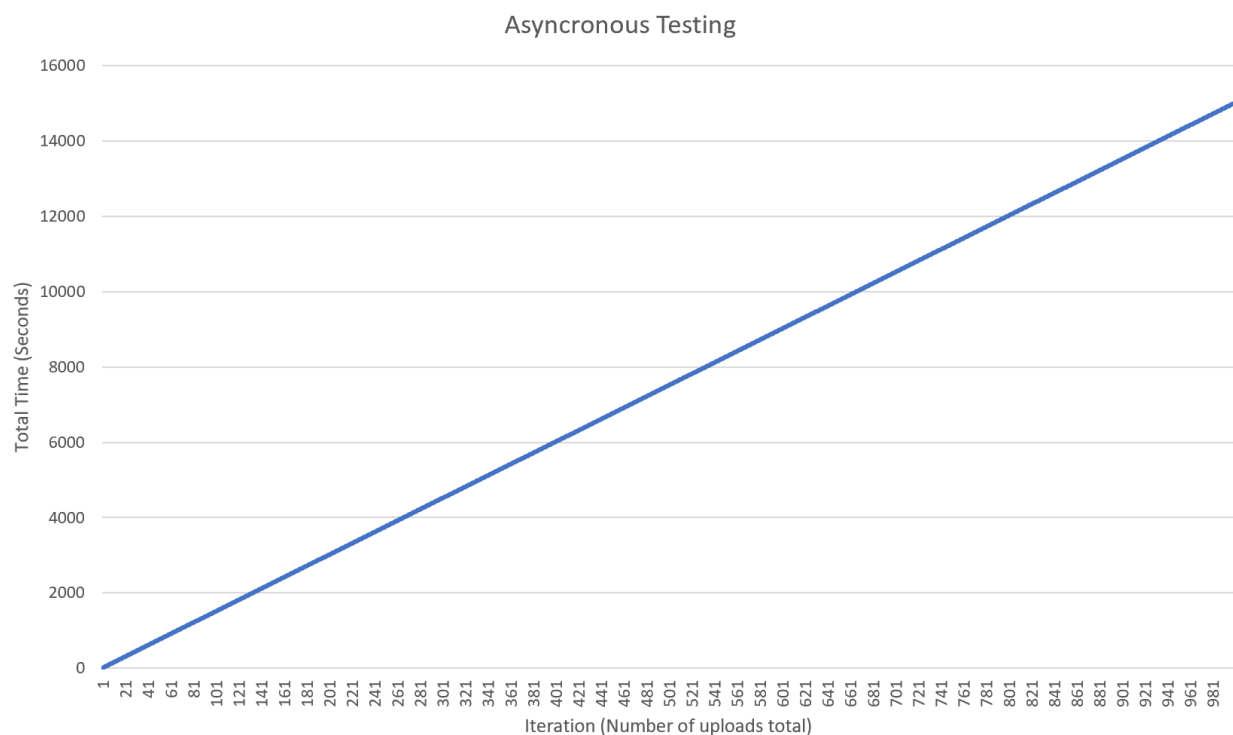


Figure 13 The result of asynchronous scalability testing

It is important to remember that blocks are mined one after another which means that no two blocks can be submitted at once. Due to this reason the Y is an accumulation of the times before it, meaning that the time needed to perform a given iteration could be expressed by 15 times the iteration needed. If this test was to be performed again it would be beneficial to perhaps increase the number to around 5,000 or even 10,000 but would presumably give the same result. Since this testing is very time consuming it is outside the scope of the project.

5.6 Access Control Performance

Test cases involving both authorized and unauthorized addresses were both used to carry out a performance analysis on the prototype. Since the overall functionality of the systems revolves around patients being able to both see and maintain their own data it is imperative to prove that this is the case. The system is designed in such a way that patients then are able to grant access to users that they choose to do so with i.e., doctors. A patient who wishes to grant access to a doctor user must first find out that user's personal ID number, this number using the blockchain returns the doctor's address and adds it to a list of authorized users who are able to access the patient's records. The doctor can then, successfully both call for previous medical records of the patient or submit their own new data for the patient's records. When this new record is added, a transaction is carried out and is verified by the miners on the network, following this the record is added to the blockchain where it is then broadcasted to all nodes on the network. Doing this a patient can then verify who is accessing or using their data. When an unauthorized user tries to call data of a user who has not authorized them, an error message is returned. Following this, a record of the unsuccessful attempt is added to the chain. In this manner the blockchain addresses issues brought up in the literature reviews regarding the control of user's data and provides an excellent countermeasure to said issues.

Following these results of rigorous testing on user access within the prototype, it is apparent that the design successfully achieves the outcome of creating a way for records to be shared among users, with an emphasis on authentication and identity. Furthermore, the smart contract-based access control is able to preserve sensitive healthcare information against attacks from outside the system. For these reasons the design goal of the system has been met.

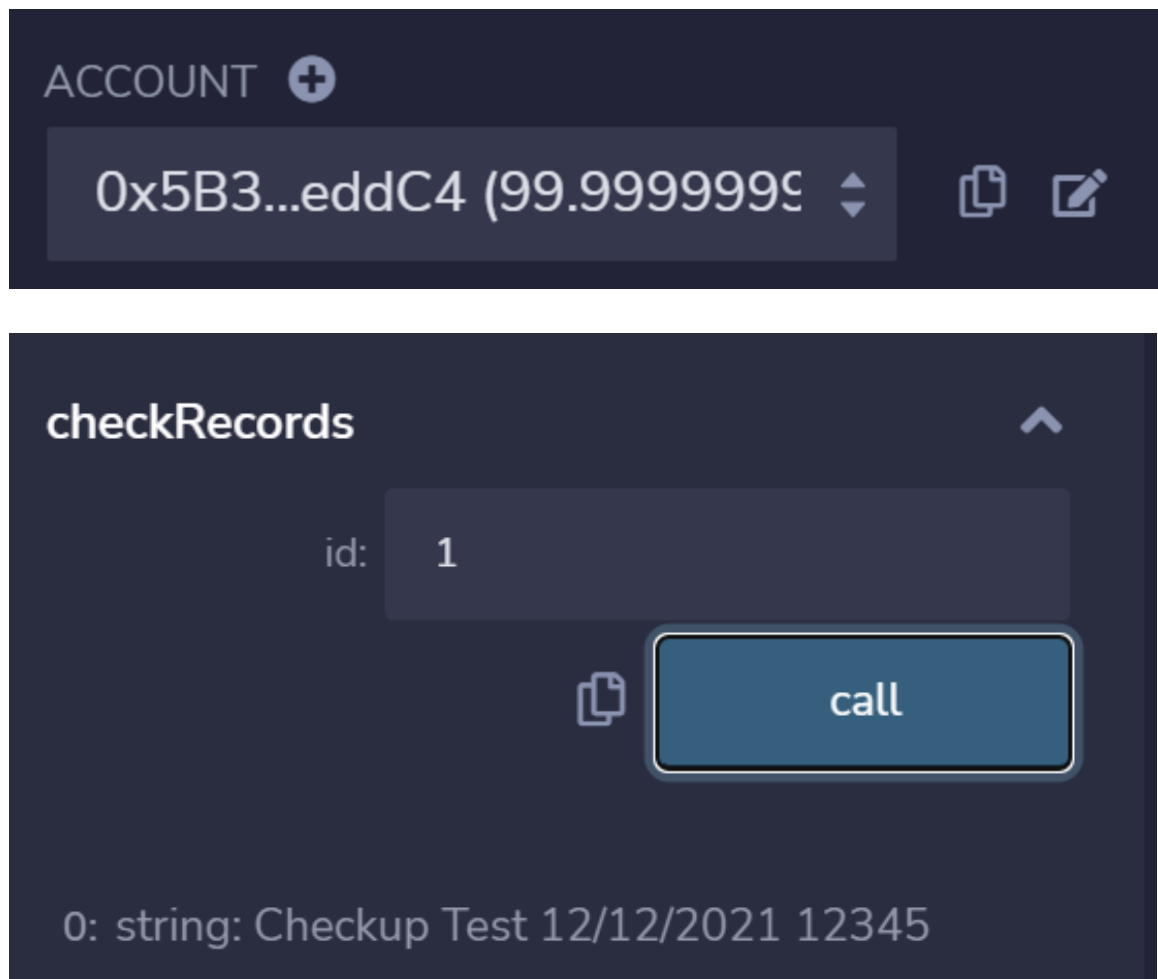


Fig. X Shows a user checking one of their records

5.7 Security Performance

Within this section different scenarios are drawn to attention that may be commonly used against a system designed to secure a user's records. The goal is to discuss these scenarios regarding the prototype and evaluate the features implanted to stop said scenarios from occurring. It is obvious that the healthcare sector is in a crisis when it comes to the security surrounding patient's medical data, so it is important countermeasures are used to help defend against potential attacks and guarantee data protection

5.7.1 Scenario 1

If a malicious user was able to access the data storage system used without the need for authentication, would this user be able to obtain and read data that they should not have access to? Within the proposed design, all the data is being encrypted using a public key, for a user's to be able to read the information off the IPFS storage system they would need the corresponding private key to do so. In this environment the private key for each user is unique and not even known by themselves and only the smart contract, therefore it would nearly be impossible for an attacker to guess the key and encrypt the data without alerting the network to the malicious intent.

5.7.2 Scenario 2:

Given that an attacker can send unauthorized transactions to request information off the framework to try and cheat their way to gaining some information that they shouldn't see, is it possible for such an attacker be able to tamper with the access controls to add themselves to the authorization for a patient? To carry out such a transaction on the network, said user needs to use their own private key (PrKey), public key (PuKey), and their request ID to carry out said transaction (Tx).

$$AuthorizedTx \leftarrow Sign_{SK} (rawTx \{PuKey, requestID, timestamp\}) \quad (10)$$

Since only the public key is known to the network, while the private key is only known by individual users, the attacker is unable to carry out the transaction, since they do not possess a private key to sign the transaction with. Furthermore, any transaction found to be invalid are removed from the network by miners. For these reasons, the system is secure against outside attacks.

5.8 Issues Found

Besides the added benefits that the technology brings to the table, are some other issues that need to be addressed and carefully thought about. Such as the following.

5.8.1 Scalability & Computing power

The solution that was developed meets the requirements set out, however there are some issues regarding the services it provides regarding to the scalability of the system. Since the healthcare sector is made up of a lot of varying and different parties the system might not necessarily keep up. This is mainly due to the fact that healthcare services often collaborate and share records. The scalability may be addressed by making use of an Internet of things (IOT) solution, in which the IoT design may be implemented to provide an interconnection web of many devices using blockchain from different parties (Novo, 2018). Furthermore, since different parties may have different roles there is a need to extend the prototype to account for people of different ranking i.e., a doctor vs nurse. The system should in essence be able to adapt and scale to different healthcare scenarios while also providing a high level of reliability. For much of the work done during this project only one or two machines were available for use, a further scope for this project would involve the use of a lot more different machines in order to simulate a much more live environment.

5.8.2 Low Latency Networks

An issue that the healthcare application faces is network latency. Sense data which can lead to a network becoming congested and may be the case with the developed framework. 'Edge Computing' offers a promising solution to such issues (Kang et al., 2018). In terms of blockchain, a lightweight design solution is a must need if there is any hope for the technology to be adapted into the healthcare sector to help fully optimize transaction as well as data processing for better latency effectiveness (Liu et al., 2019).

5.8.3 Data Uploading Issues

The uploading of information to the proposed storage does however raise some concerns for the blockchain. Since an off-chain solution is still being used, the server can still become 'curious' as discussed during the literature review. This also extends to the miners on the network, who could potentially infer personal information from nodes and blocks by recognizing patterns. For such situations, there are some strategies that show promise when it comes to these issues. Reinforced learning is once such techniques that could solve for these concerns in a healthcare setting (Min et al., 2018).

5.9 Results

The use of various smart contracts were both designed and implemented for the application in order to test the secureness of housing medical records using blockchain based off of concerns brought up in the reviews. Testing and evaluation was then carried out by the Ethereum blockchain. Furthermore, several tests were run to calculate the costs associated to discern the economic viability of said framework and the work carried out shows that it can. As compared to the literature review, the costs associated with using said system were drastically lower than see discussed in section 2.7.3. Experiments and testing were also carried out to find out the execution times involved with some of the more important functions within the system. This allowed for an evaluation to be carried out to gauge the usability of said system and find out just how secure records would be. The system showed it was quick and effective to use as the whole process fell within the average time frame seen with other systems of the same kind i.e., 15 seconds to validate and process. The system showed resilience against all of the most common attackers that were researched and proves to be an effective tool for the storage of medical records. Thus, this prototype allows for all patient records to be immutable, tamper proof, authentic, resilient, and allow for easy sharing to doctors.

6 Discussions

Overall, the system at hand allows for a relatively cheap way for the storage and sharing of medical records. However, since a live system does involve currency in the form of Ether to work it does ask the question of who should cover the cost of sharing data. Future work regarding the implementation of Blockchain in the healthcare sector would need to be carried out to discover the costs of not just running the functions and contracts but administrative fees, node management, computers etc. One would guess that the fees incurred would probably be covered by the patient themselves in the form of medical expenses. Furthermore, since miners are being used to validate transactions, would fellow patients be used and given passive incentives, and would this cover the costs? Additionally, it was found that by itself the blockchain technology itself was not an end all solution to issues of privacy, it is only through the combination of privacy as well as encryption techniques such as the ones discussed in section 3.4. As mentioned,

depending on blockchain to help secure off-chain medical information is impracticable. Therefore, to secure a healthcare system, proper cryptographic primitives are needed to achieve, integrity, confidentiality, privacy protection, and access control. IBE, ABE, and PRE are widely adapted tools, and a blend of the different techniques are used to help ensure strict encryption with a combination of ECC and ECQV. It can be then expected that similar cryptography will be used to play an important role in blockchain-based information sharing platforms such as the proposed prototype. Within the prototype a mixture of the encryption types of previously mention be crucial when it comes to the design and implementation of the applications to ensure the security and privacy of patient's data.

Overall, the system is secured against the most common attacks. However, due to the growing rate of the technology it is impossible to test for every type of method possible and as such the most important thing to do when developing a system like this is to keep an ear to the ground and know the field. Keeping the versions of the technologies up to date and future development would ensure the solution is future proof but there is always a chance a person could discover a back door to the used Blockchain technologies.

7 Conclusion & Future Work

To resolve the bulky task of storing medical records of patients' this piece of work proposes a novel medical record sharing scheme that is enabled by the use of the blockchain technology. The different challenges that current the healthcare-based systems face were identified and noted as goals for said proposal to overcome. Within this work a focus is made on designing a system with secure access control mechanisms by making use of various different smart contracts using the Ethereum blockchain.

An investigation was made into the performance of the proposed approaches and a various smart contracts were deployed to the Rinkeby testing network for evaluation purposes. The integration of the IPFS peer-to-peer was crucial for the prototype to function as intended to achieve a decentralized data sharing and storing. The result of testing shows that the framework developed can allows for patients to share their medical records over the application with specified doctors in a reliable and safe manner compared to traditional systems. In particular, the system is able

to detect malicious users and subsequently prevent them access to the system, aiming for a heightened level of patient privacy and network protection. Extensive security analysis on various technical aspects of the application was carried out showing the advantage that the proposal has over more conventional solutions for storing data. Based on the achievements of the framework, it is believed that with additional work carried out that the blockchain based model might be a step in the right direction towards an effective way to manage healthcare records, which could be promising in many different applications in the sector. On top of this, testing has shown regarding the computing costs of the system that the solution is economically viable.

However, at time it was found that working the technology in such a way that it was not originally intended was rather cumbersome. Furthermore, the work carried out was found to be vague when it comes to the full scope of the healthcare secure. In the future the development of a blockchain that is able to store all relevant data to itself rather than an off-chain component would be of interest as well as a more user-friendly way to include inputs rather than the additional development of a front-end JavaScript user interface. Since the technology is still rather in its infancy it was found that not a lot of flexibility when it came to designing was found, additionally all documentation being the same on certain subjects meant that most times there was only one solution to any issue that was ran into such as creating the many functions within the smart contracts.

Since this work looked into a hybrid of both research and development, it would be interesting to carry out additional work now that a good grasp around what needs to be done is obtained. This work would greatly expand on the work done in this paper and aim to create and more robust system that is applicable to more situation than just a patient sharing their records with doctor. Future work into development of a Blockchain that isn't Ethereum, but something better suited to the needs of a healthcare environment would also be of great interest. This would allow for greater customization when it comes to the overall design of the system and addresses the issues regarding ease of use.

In future work to maintain a high standard of healthcare services, blockchain related application may require some guarantees when it comes to the quality of the service provided. For instance, clinical support platforms can make use of things such as machine learning to analyze medical records to predict future health related issues when it comes to patients (Nguyen et al., 2019). This could assist providers in making diagnoses, while also preserving data privacy and security.

Overall, the work has shown that there would be significant benefits if blockchain were to be used in the healthcare field regarding the storage of records. This is shown through the performance of the prototype as well as the cost-based analysis which both show feasibility for a live system. However, due to some of the short comings of the technology regarding scalability, skill storages, and a social climate that is slow to change the norm as seen with the literature. One would assume that the technology should be given time to mature before it is properly sought after to solve for privacy / security concerns that have been mentioned all throughout this work. For this reason, it can be said, with both time and future research and development, the technology could be seen in the not-so-distant future as the industry standard for housing and securing personal medical records.

9 References

- Aitzhan, N., Z., Svetinovic, D. (2018). *Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams*. Available at: <<https://ieeexplore.ieee.org/abstract/document/7589035>>
- Alhadhrami, Z., Alghfeli, S., Alghfeli, M., Abedlla, J. A., & Shuaib, K. (2017). Introducing blockchains for healthcare. In *2017 international conference on electrical and computing technologies and applications (ICECTA)* (pp. 1-4). IEEE.
- Alonso, S. G., Arambarri, J., López-Coronado, M., & de la Torre Díez, I. (2019). Proposing new blockchain challenges in ehealth. *Journal of medical systems*, 43(3), 64.
- Anderson, J. (2018). Securing, standardizing, and simplifying electronic health record audit logs through permissioned blockchain technology.
- Apostolaki, M., Maire, C., and Vanbever, L. (2021) Nsg.ee.ethz.ch. [online] Available at: https://nsg.ee.ethz.ch/fileadmin/user_upload/publications/fc21final97.pdf
- Atzei, N., Bartoletti, M., & Cimoli, T. (2016). A survey of attacks on Ethereum smart contracts (No. 1007).
- Azaria, A., Ekblaw, A., Vieira, T., & Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In *2016 2nd international conference on open and big data (OBD)* (pp. 25-30). IEEE.
- Badr, S., Gomaa, I., & Abd-Elrahman, E. (2018). Multi-tier blockchain framework for IoT-EHRs systems. *Procedia Computer Science*, 141, 159-166.
- Bit Degree. (2021). EOS vs Ethereum: is EOS a Better Ethereum, from <https://www.bitdegree.org/crypto/tutorials/eos-vs-ethereum>
- Böhme, R., Christin, N., Edelman, B., & Moore, T. (2015). Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2), 213-38.

- Boneh, D., & Franklin, M. (2001). Identity-based encryption from the Weil pairing. In *Annual international cryptology conference* (pp. 213-229). Springer, Berlin, Heidelberg.
- Burniske, C., Vaughn, E., Shelton, J., & Cahana, A. (2016). How blockchain technology can enhance EHR operability. *Gem| Ark Invest Res.*
- Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *white paper*, 3(37).
- Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *white paper*, 3(37).
- Cachin, C., & Vukolić, M. (2017). Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*.
- California Health Care Foundation (CHCF). (2005). Clinical Data Standards in Health Care: Five Case Studies. Available from: <http://www.chcf.org/publications/2005/07/clinical-data-standards-in-healthcare-five-case-studies>.
- Cardano. (2021). From <https://cardano.org/>
- Castro, M., & Liskov, B. (1999). Practical byzantine fault tolerance. In *OSDI* (Vol. 99, No. 1999, pp. 173-186).
- Chen, H., Pendleton, M., Njilla, L., & Xu, S. (2019). A survey on Ethereum systems security: Vulnerabilities, attacks, and defenses. *arXiv preprint arXiv:1908.04507*.
- Chen, H., Pendleton, M., Njilla, L., & Xu, S. (2020). A survey on Ethereum systems security: Vulnerabilities, attacks, and defenses. *ACM Computing Surveys (CSUR)*, 53(3), 1-43.
- Cloud Security Alliance. (2021). Can Blockchains Survive the Quantum Computer? From <https://cloudsecurityalliance.org/blog/2021/02/09/can-blockchains-survive-the-quantum-computer/>

CoinMarketCap. (2021). Cardano price today, ADA to USD live, marketcap and chart. From <https://coinmarketcap.com/currencies/cardano/>

CoinMarketCap. (2021). Cardano price today, ADA to USD live, marketcap and chart. From <https://coinmarketcap.com/currencies/cardano/>

CoinMarketCap. (2021). Ethereum price today, ETH to USD live, marketcap and chart. From <https://coinmarketcap.com/currencies/ethereum/>

Confais, B., Lebre, A., & Parrein, B. (2017, May). An object store service for a fog/edge computing infrastructure based on ipfs and a scale-out nas. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)* (pp. 41-50). IEEE.

Crypto. (2021). Node.js v16.6.1 Documentation. From https://nodejs.org/api/crypto.html#crypto_crypto.

Cryptonews. (2021). Why We Are Building on Tezos, Consortium Comprising BMW, Audi and Others Reveal. From <https://cryptonews.net/en/news/altcoins/317465/>

Cryptopedia. (2021). Ethereum 2.0: Ethereum Blockchain Scaling Solution, Gemini. From <https://www.gemini.com/cryptopedia/Ethereum-2-0-proof-of-stake-pos-blockchain-serenity#section-new-tools-to-boost-Ethereum-blockchain-throughput>

Dagher, G. G., Mohler, J., Milojkovic, M., & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable cities and society*, 39, 283-297.

Dagher, G. G., Mohler, J., Milojkovic, M., & Marella, P. B. (2018). Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. *Sustainable cities and society*, 39, 283-297.

Digiconomist. (2021). Ethereum Energy Consumption Index from <https://digiconomist.net/Ethereum-energy-consumption>

- Dwivedi, A. D., Srivastava, G., Dhar, S., & Singh, R. (2019). A decentralized privacy-preserving healthcare blockchain for IoT. *Sensors*, 19(2), 326.
- Dwyer, G. P. (2015). The economics of Bitcoin and similar private digital currencies. *Journal of financial stability*, 17, 81-91.
- Eciesjs. (2021)., from <https://www.npmjs.com/package/eciesjs>
- Edureka. (2021) What is Hyperledger | Introduction to Hyperledger Project. From <https://www.edureka.co/blog/what-is-hyperledger/>
- EOS, (2021). From <https://eos.io/>
- Esposito, C., De Santis, A., Tortora, G., Chang, H., & Choo, K. K. R. (2018). Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1), 31-37.
- Ethereum. (2018). *Blockchain App Platform*. Available: <https://www.Ethereum.org.Reference>
- Fan, K., Wang, S., Ren, Y., Li, H., & Yang, Y. (2018). Medblock: Efficient and secure medical data sharing via blockchain. *Journal of medical systems*, 42(8), 1-11.
- Farouk, A., Alahmadi, A., Ghose, S., & Mashatan, A. (2020). Blockchain platform for industrial healthcare: Vision and future opportunities. *Computer Communications*, 154, 223-235.
- Fiat A., Naor M. (1994) Broadcast Encryption. In: Stinson D.R. (eds) *Advances in Cryptology — CRYPTO' 93*. CRYPTO 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48329-2_40
- Franceschi, M. (2019) “ComeHere: Exploiting Ethereum for secure sharing of health-care data,” in Proc. Eur. Conf. Parallel Process, pp. 585–596.
- Google. (2021). The Go Programming Language. From <https://go.dev/doc/>
- Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (pp. 89-98).

- Guo, R., Shi, H., Zhao, Q., & Zheng, D. (2018). Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE access*, 6, 11676-11686.
- Gwyneth Iredale. (2019). 101 Blockchains. 2021. Introduction to Permissioned Blockchains. Available at: <https://101blockchains.com/permissioned-blockchain/#:~:text=The%20key%20permissioned%20blockchains%20include%20Quorum%2C%20Corda%2C%20and%20Hyperledger%20Fabric.>
- Harman, L. B. (2006). *Ethical challenges in the management of health information*. Jones & Bartlett Learning.
- Hassan, M. M., Ali, M. A., Bhuiyan, T., Sharif, M. H., & Biswas, S. Quantitative Assessment on Broken Access Control Vulnerability in Web Applications. In International Conference on Cyber Security and Computer Science 2018.
- Hcltech. (2021). *Byzantine Fault Tolerance (BFT) and its significance in Blockchain world | HCL Blogs*. Available at: <<https://www.hcltech.com/blogs/byzantine-fault-tolerance-bft-and-its-significance-blockchain-world#:~:text=Byzantine%20fault%20tolerance%20is%2050,as%20network%20latency%20approaches%20infinity.>>>
- Health Information and Quality Authority (HIQA). (2017). Overview of healthcare interoperability standards, <https://www.hiqa.ie/sites/default/files/2017-01/Healthcare-Interoperability-Standards.pdf>
- Hölbl, M., Kompara, M., Kamišalić, A., & Nemec Zlatolas, L. (2018). A systematic review of the use of blockchain in healthcare. *Symmetry*, 10(10), 470.
- Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., ... & Liu, Y. (2021). Transaction-based classification and detection approach for Ethereum smart contract. *Information Processing & Management*, 58(2), 102462.
- Humbeeck, A. V. (2019). The blockchain-GDPR paradox. *Journal of Data Protection & Privacy*.

Hyla, T., & Pejaś, J. (2019). eHealth integrity model based on permissioned blockchain. *Future Internet*, 11(3), 76.

Hyperledger Foundation. (2022). From <https://www.hyperledger.org/about>

Hyperledger. (2021). *Hyperledger – Open Source Blockchain Technologies*. Available at: <https://www.hyperledger.org/>

IBM. (2018). Trust in trade, 2018, Available: <https://public.dhe.ibm.com/common/ssi/ecmlgb/en/gbe03771usen/global-business-services-global-business-services-executive-brief-{}gbe03771usen-20180529pdf>.

Infura. (2021). Ethereum API, IPFS API, & Gateway. ETH Nodes as a Service from <https://infura.io/>

Investopedia. (2021). *Consensus Mechanism (Cryptocurrency)*. Available at: <https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp#:~:text=A%20consensus%20mechanism%20is%20a,systems%2C%20such%20as%20with%20cryptocurrencies>.

Investopedia. (2021). *Proof of Activity*. Available at: [https://www.investopedia.com/terms/p/proof-activity-cryptocurrency.asp#:~:text=Proof%2Dof%2Dactivity%20\(PoA\)%20is%20a%20blockchain%20consensus,miners%20arrive%20at%20a%20consensus.>](https://www.investopedia.com/terms/p/proof-activity-cryptocurrency.asp#:~:text=Proof%2Dof%2Dactivity%20(PoA)%20is%20a%20blockchain%20consensus,miners%20arrive%20at%20a%20consensus.>)

Ismail, L., Materwala, H., & Zeadally, S. (2019). Lightweight blockchain for healthcare. *IEEE Access*, 7, 149935-149951.

Jamshed, N., Ozair, F., Sharma, A., and Aggarwal, P. (2015) "Ethical issues in electronic health records: A general overview", *Perspect. Clin. Res*, vol. 6, no. 2, pp. 73.

Jin, H., Luo, Y., Li, P., & Mathew, J. (2019). A review of secure and privacy-preserving medical data sharing. *IEEE Access*, 7, 61656-61669.

- Kang, J., Yu, R., Huang, X., Wu, M., Maharjan, S., Xie, S., & Zhang, Y. (2018). Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, 6(3), 4660-4670.
- Kaur, H., Alam, M. A., Jameel, R., Mourya, A. K., & Chang, V. (2018). A proposed solution and future direction for blockchain-based heterogeneous medicare data in cloud environment. *Journal of medical systems*, 42(8), 1-11.
- Kemkarl, O. S., & Dahikar, D. P. B. (2012). Can electronic medical record systems transform health care? potential health benefits, savings, and cost using latest advancements in ict for better interactive healthcare learning. *International Journal of Computer Science & Communication Networks*, 2(3/6), 453-455.
- Kevin McAvey. (2021). "Investing in The Data Systems We Need to Create The Health System We Deserve, " Health Affairs Blog, June 4, 2021.
- Klent. (2021) Cardano's Rumored Smart Contract Issues and Limitations Thoroughly Debunked. From <https://zycrypto.com/cardanos-rumored-smart-contract-issues-and-limitations-thoroughly-debunked/>
- Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association*, 24(6), 1211-1220.
- La Capra. (2021) Is EOS Dead? What Happened to the Ethereum Killer? From <https://coinmarketcap.com/alexandria/article/is-eos-dead-what-happened-to-the-ethereum-killer>
- Labs, P., (2021). *IPFS Powers the Distributed Web*. [online] IPFS. Available at: <<https://ipfs.io/>>
- Laxmeshwar, S., & Deepak, N. R. (2020). A Survey on Efficient Block Chain Authentication Scheme Based on Electronic Health Records.

- Lee, S. J., Cho, G. Y., Ikeno, F., & Lee, T. R. (2018). BAQALC: blockchain applied lossless efficient transmission of DNA sequencing data for next generation medical informatics. *applied sciences*, 8(9), 1471.
- Lewko, A., & Waters, B. (2011). Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 568-588). Springer, Berlin, Heidelberg.
- Li, X., Huang, X., Li, C., Yu, R., & Shu, L. (2019). EdgeCare: leveraging edge computing for collaborative data management in mobile healthcare systems. *IEEE Access*, 7, 22011-22025.
- Liljeqvist. (2021). From <https://academy.moralis.io/blog/smart-contract-platforms-eos-vs-ethereum-vs-rsk-vs-ardano#:~:text=Ethereum%20was%20developed%20and%20designed,scalability%2C%20speed%2C%20and%20flexibility>.
- Lin, I. C., & Liao, T. C. (2017). A survey of blockchain security issues and challenges. *Int. J. Netw. Secur.*, 19(5), 653-659.
- Liu, Y., Wang, K., Lin, Y., & Xu, W. (2019). $\mathsf{LightChain}$: a lightweight blockchain system for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 15(6), 3571-3581.
- Lv, P., Wang, L., Zhu, H., Deng, W., & Gu, L. (2019). An IoT-oriented privacy-preserving publish/subscribe model over blockchains. *IEEE Access*, 7, 41309-41314.
- Maharjan, Pradip Singh, "Performance Analysis of Blockchain Platforms" (2018). UNLV Theses, Dissertations, Professional Papers, and Capstones. 3367. <http://dx.doi.org/10.34917/14139888>
- Mahut. (2021). eUTXO transaction model Vs Account Based Transaction Model. From <https://cardano.stackexchange.com/questions/696/eutxo-transaction-model-vs-account-based-transaction-model>

- Mamoshina, P., Ojomoko, L., Yanovich, Y., Ostrovski, A., Botezatu, A., Prikhodko, P., ... & Zhavoronkov, A. (2018). Converging blockchain and next-generation artificial intelligence technologies to decentralize and accelerate biomedical research and healthcare. *Oncotarget*, 9(5), 5665.
- Medicalchain. (2022). From <https://medicalchain.com/en/team/>
- Metamask. (2021). *MetaMask documentation*. Available at: <<https://metamask.io/>>
- Michelson. (2021). The Language of Tezos. From <https://www.michelson.org/>
- Microsoft. (2021). Retrieved 21 July 2021, from <https://www.microsoft.com/en-us/research/wp-content/uploads/2002/01/IPTPS2002.pdf>
- Min, M., Wan, X., Xiao, L., Chen, Y., Xia, M., Wu, D., & Dai, H. (2018). Learning-based privacy-aware offloading for healthcare IoT with energy harvesting. *IEEE Internet of Things Journal*, 6(3), 4307-4316.
- M. Morris. (2016). Global health care outlook: Battling costs while improving care," Deloitte.
- Mobi Health News*. (2018). *Survey: 90 Percent of Healthcare Organizations use or Plan to use Mobile Devices*, Available: <https://www.mobihealthnews.com/content/survey-90-percent-healthcare-organizations-use-or-plan-use-mobile-devices>.
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.
- Next.js. (2021). Vercel - The React Framework. From <https://nextjs.org/>
- Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2019). Blockchain for secure ehrs sharing of mobile cloud-based e-health systems. *IEEE access*, 7, 66792-66806.
- Novo, N. (2018). "Blockchain meets IoT: An architecture for scalable access management in IoT", *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1184-1195.

- Omar, A., Bhuiyan, M. Z. A., Basu, A., Kiyomoto, S., & Rahman, M. S. (2019). Privacy-friendly platform for healthcare data in cloud based on blockchain environment. *Future generation computer systems*, 95, 511-521.
- ONA General Counsel. (2016). Members and patient privacy be aware and beware! From <https://www.ona.org/>
- Onik, M. M. H., Aich, S., Yang, J., Kim, C. S., & Kim, H. C. (2019). Blockchain in healthcare: Challenges and solutions. In *Big data analytics for intelligent healthcare management* (pp. 197-226). Academic Press.
- OWASP. (2017). The Open Web Application Security Project From <https://www.veracode.com/directory/owasp-top-10>
- P. A. I. (2015). How to reveal application vulnerabilities? SAST, DAST, IAST and others. *Positive Technologies*.
- Pfitzmann, A., & Hansen, M. (2010). A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management.
- Pilkington, M. (2016). Blockchain technology: principles and applications. In *Research handbook on digital transformations*. Edward Elgar Publishing.
- Poh, G. S., Chin, J. J., Yau, W. C., Choo, K. K. R., & Mohamad, M. S. (2017). Searchable symmetric encryption: designs and challenges. *ACM Computing Surveys (CSUR)*, 50(3), 1-37.
- Popa, R. A., Lorch, J. R., Molnar, D., Wang, H. J., & Zhuang, L. (2011). Enabling Security in Cloud Storage SLAs with CloudProof. In *USENIX Annual Technical Conference* (Vol. 242, pp. 355-368).
- Poston, H. (2020). Mapping the OWASP top ten to blockchain. *Procedia Computer Science*, 177, 613-617.

- Praitheeshan, P., Pan, L., Yu, J., Liu, J., & Doss, R. (2019). Security analysis methods on ethereum smart contract vulnerabilities: a survey. *arXiv preprint arXiv:1908.08605*.
- Praitheeshan, P., Pan, L., Yu, J., Liu, J., & Doss, R. (2019). Security analysis methods on Ethereum smart contract vulnerabilities: a survey. *arXiv preprint arXiv:1908.08605*.
- Quaini, T., Roehrs, A., da Costa, C. A., & da Rosa Righi, R. (2018). A MODEL FOR BLOCKCHAIN-BASED DISTRIBUTED ELECTRONIC HEALTH RECORDS. *IADIS International Journal on WWW/Internet*, 16(2).
- Quaini, T., Roehrs, A., da Costa, C. A., & da Rosa Righi, R. (2018). A MODEL FOR BLOCKCHAIN-BASED DISTRIBUTED ELECTRONIC HEALTH RECORDS. *IADIS International Journal on WWW/Internet*, 16(2).
- React. (2021). A JavaScript library for building user interfaces. From <https://reactjs.org/>
- Remix. (2021). Ethereum IDE documentation. From <https://remix.ethereum.org/>.
- Rinkeby. (2021) Network Dashboard. From <https://www.rinkeby.io/#stats>.
- RSK, (2021). From <https://www.rsk.co/>
- Sahai, A., & Waters, B. (2005). Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 457-473). Springer, Berlin, Heidelberg.
- Sanchez. (2021). Cardano's Extended UTXO accounting model – built to support multi-assets and smart contracts - IOHK Blog. From <https://iohk.io/en/blog/posts/2021/03/11/cardanos-extended-utxo-accounting-model/>
- Satoshi, N. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*, Available: <https://bitcoin.org/bitcoin.pdf>.
- Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8, 24416-24427.

- Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8, 24416-24427.
- Schnitzbauer, M. (2021). Smart Contracts in Healthcare. *Digitalization in Healthcare*, 211.
- Seng, L. K., Ithnin, N., & Said, S. Z. M. (2018). The approaches to quantify web application security scanners quality: a review. *International Journal of Advanced Computer Research*, 8(38), 285-312.
- Shamir, A. (1984, August). Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques* (pp. 47-53). Springer, Berlin, Heidelberg.
- Shen, B., Guo, J., & Yang, Y. (2019). MedChain: Efficient healthcare data sharing via blockchain. *Applied sciences*, 9(6), 1207.
- Shi, S., He, D., Li, L., Kumar, N., Khan, M. K., & Choo, K. R. (2020). Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. *Computers & security*, 97, 101966. <https://doi.org/10.1016/j.cose.2020.101966>
- Shrimpy. (2021). What Is Cardano (ADA)? From <https://academy.shrimpy.io/post/what-is-cardano-ada>
- Sinclair. (2021) From <https://www.coindesk.com/tech/2021/09/12/cardano-gains-smart-contract-capability-following-alonzo-hard-fork/>
- Solidity. (2021). Docs.soliditylang.org. — *Solidity 0.7.4 documentation*. Available at: <<https://docs.soliditylang.org/en/v0.7.4/>>
- Statista. (2021). Countries with highest primary energy consumption. Available at: <<https://www.statista.com/statistics/263455/primary-energy-consumption-of-selected-countries/>>
- Swan, M. (2015). Blockchain: Blueprint for a new economy, O'Reilly Media Inc.

- Szabo, N. (1996). "Smart contracts: Building blocks for digital markets" in EX-TROPY J. Transhumanist Thought, no. 16.
- Tezos. (2021). Welcome to the Tezos Agora Wiki! From <https://wiki.tezosagora.org/>
- The Irish Times. (2021). HSE still facing IT system difficulties 20 days after cyberattack. Available from <https://www.irishtimes.com/news/ireland/irish-news/hse-still-facing-it-system-difficulties-20-days-after-cyberattack-1.4581935>
- Tian, F. (2016, June). An agri-food supply chain traceability system for China based on RFID & blockchain technology. In *2016 13th international conference on service systems and service management (ICSSSM)* (pp. 1-6). IEEE.
- Uddin, M. A., Stranieri, A., Gondal, I., & Balasubramanian, V. (2018). Continuous patient monitoring with a patient centric agent: A block architecture. *IEEE Access*, 6, 32700-32726.
- Velmovitsky, P. E., Bublitz, F. M., Fadrique, L. X., & Morita, P. P. (2021). Blockchain Applications in Health Care and Public Health: Increased Transparency. *JMIR Medical Informatics*, 9(6), e20713.
- Wang, S., Zhang, Y., & Zhang, Y. (2018). A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *Ieee Access*, 6, 38437-38450.
- Washington, L. C. (2008). *Elliptic curves: number theory and cryptography*. CRC press.
- web3.js (2021). Ethereum JavaScript API — web3.js 1.0.0 documentation. From <https://web3js.readthedocs.io/en/v1.4.0/>
- Wikipedia. (2021). En.wikipedia.org *Merkle tree* Available at: <https://en.wikipedia.org/wiki/Merkle_tree#/media/File:Hash_Tree.svg>
- Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014), 1-32.

- World Economic Forum. (2015). "Deep Shift. Technology Tipping Points and Societal Impact, Available from http://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_report_2015.pdf.*
- Xia, Q., Sifah, E. B., Smahi, A., Amofa, S., & Zhang, X. (2017). BBDS: Blockchain-based data sharing for electronic medical records in cloud environments. *Information*, 8(2), 44.
- Xu, X., Sun, G., Luo, L., Cao, H., Yu, H., & Vasilakos, A. V. (2021). Latency performance modeling and analysis for hyperledger fabric blockchain network. *Information Processing & Management*, 58(1), 102436.
- Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017, April). A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE international conference on software architecture (ICSA)* (pp. 243-252). IEEE.
- Yang, H., & Yang, B. (2017). A blockchain-based approach to the secure sharing of healthcare data. *Nisk Journal*, 100-111.
- Ying, Z., Wei, L., Li, Q., Liu, X., & Cui, J. (2018). A lightweight policy preserving EHR sharing scheme in the cloud. *IEEE Access*, 6, 53698-53708.
- Zhang, A., & Lin, X. (2018). Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *Journal of medical systems*, 42(8), 1-18.
- Zhang, J., Xue, N., & Huang, X. (2016). A secure system for pervasive social network-based healthcare. *Ieee Access*, 4, 9239-9250.
- Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning-based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1), 1-38.
- Zhao, Q., Chen, S., Liu, Z., Baker, T., & Zhang, Y. (2020). Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems. *Information Processing & Management*, 57(6), 102355.

Zheng, X., Sun, S., Mukkamala, R. R., Vatrappu, R., & Ordieres-Meré, J. (2019). Accelerating health data sharing: A solution based on the internet of things and distributed ledger technologies. *Journal of medical Internet research*, 21(6), e13583.

10 Appendices

10.1 Appendix 1. List of Notations

Notation	Description
(d, Q)	Private & public key pair
M	Data in plaintext
M_e	Encrypted data
$\{M\}_d$	Digital signature for the message M , using the private key d

$H(M)$	Hashed M , using a form of hashing like MD or more likely SHA256
$E_Q(M)$	Encrypted M , by using Q results in M_e
$D_d(M_e)$	Decryption of M , by using the private key d which results in M
ID_u	unique Identifier of a given user
$cert$	Digital certificate
$data_i$	Data of a user
$data_e$	Encrypted data of a user
$address$ storage	The address of the location that the data is being stored within the file storage
OH	The original hash of data

10.2 Appendix 2. Elliptic-curve cryptography (ECC)

Within this section the preliminaries of ECC are discussed. This allows the development of the necessary background information to understand how ECC works and the role it plays within the discussed prototype (Washington, 2008).

ECC, which allows for a lightweight public key distribution is based off the algebraic structure of Elliptic curves (EC) (Washington, 2008). A curve of finite fields F_p is referred to as $E_p(a, b)$ and is then defined as:

$$y^2 = x^3 + ax + b \quad (11)$$

where a & b are two constants seen in f_p & $\Delta = 4a^3 + 27b^2 \neq 0$. The foundation for generating the points of $E_p(a, b)$ of prime order q is denoted by G . The main operating functions in ECC are:

- Addition: By using addition, $P1 + P2 = R$, will then result in finding a new EC point which is denoted by R

- Multiplication: By using scalar multiplication of $e \in \mathbb{Z}$ for any given EC point P shown as $R = rP = (R_x, R_y)$, with $R_x, R_y \in F_p$ results in the point R of the RC

The privacy that ECC can provide is dependent on the following computationally problems:

- Elliptic curve Discrete Logarithm Problem (ECDLP): Within this problem, given that two points R & Q of $Ep(a, b)$, it should theoretically be almost impossible for any polynomial time-based algorithm to figure out a parameter $x \in F_q^*$, so that $Q = xR$
- Elliptic Curve Diffie Hellman Problem (ECDHP): Within this problem, given two points $R = xP, Q = yP$ with unidentified variables $x, y \in F_q^*$, it is almost impossible to then solve for a polynomial-time based algorithm that can discover the EC point xyP

10.2.1 Encryption & Decryption

By making use of ECDHP, the fashioning of a secret key to then be used by both a sender and receiver is easy. Given that (ds, Qs) and (dr, Qr) symbolizes the key pairs of both the sender and the receiver in each case (Washington, 2008). A key that is the shared K at a timestamp TS then equates to the following

$$K = H(dsQr || TS) = H(drQr || TS) \quad (12)$$

This equation then can only be derived by the sender or receiver. A message M using key K is then used to obtain a give ciphertext C by using the following

$$C = Ek(M) \quad (13)$$

The Elliptic Curve Integrated Encryption (ECIES) is very efficient when a sender wishes to use their public key for encryption without authentication (Washington, 2008). To perform this, the sender selects a random value r and calculates for $U = rG$. The key for encryption k is defined as $K =$

$H(rQR)$. The message that is to be sent consists of $C = Ek(M)$ along with a random EC point known as U .

10.2.2 Signature Creation & Verification

Using the previously mentioned key pairs (ds, Qs) and (dr, Qr) the ‘Schnorr’ signature generation can take place by using S by IDs on top of the message M and requires the following steps to occur

- The selection of a random value r and using this value to compute for $R = rG$
- Computing for $h = H(M || R)$ along with $s = r - hds$

The tuple (M, h, s) is then received by the receiver. This can then be used to verify for authenticity by carrying out the following steps

- Computing: $R = sG + hQ_s$
- Checking that: $H(M || R) == h$

10.3 Appendix 3. Elliptic Curve Qu Vanstone (ECQV)

To be able to make use of ECC for a public based key system for encryption and signatures the sender along with the receiver make use of both a public and private key pairing which can be denoted by (d, Q) , where:

$$Q = dG \quad (14)$$

G represents the variable that represents the point of the curve that was generated for use with the encryption (Washington, 2008). To prove the identity of any user using this system a certificate is needed to associate the public key with its owner. ECQV offers a lightweight solution and can be used when setting up users in a system as an effective way to provide public and private key pairing for added security. In this way it is impossible for anyone to discern the private key of a different user since the certificates only identify the public key of a user. Within the proposed prototype the key pair of an administrative user can be denoted as (d_{admin}, Q_{admin}) .

To then derive the pairing (du, Qu) for a user with the identity IDu can be written as seen in fig. X. H here is representing a hash function, this function could be SHA2 or something along the lines of SHAKE128 (M, 256), having a 256-bit output and 128-bit overall privacy. A user can figure out the public key of a different user by using IDu given the said users certificate $cert_u$ by using the following:

$$Q_u = H(cert_u \parallel ID_u)cert_u + Q_{admin}.$$

User Requesting Key Material

Administrative Body

Choose $r_u \in_R \mathbb{F}_p, R_u = r_u G$

$\xrightarrow{IDn \parallel R_n}$

$$rt \in_R \mathbb{F}_p, R_T = r_T G$$

$$Cert_u = R_u + R_T$$

$$r = H(cert_u \parallel ID_u)r_T + d_{admin}$$

$$rt \in_R \mathbb{F}_p, R_T = r_T G$$

$$\leftarrow r \parallel cert_u$$

$$d_u = H(cert_u \parallel ID_u)r_u + r$$

$$Q_u = d_u G$$

Fig. X. Steps & computing the ECQV certificate-based key construction

This is all based on the facts that

$$\begin{aligned} Q_u &= d_u G = (H(cert_u \parallel ID_u)r_u + r) G \\ &= H(cert_u \parallel ID_u)r_u G + rG \\ &= H(cert_u \parallel ID_u)R_u + (H(cert_u \parallel ID_u)r_T + d_{admin}) G \end{aligned}$$

$$\begin{aligned}
&= H(cert_u \parallel ID_u)R_u + H(cert_u \parallel ID_u)R_T + Q_{admin} \\
&= H(cert_u \parallel ID_u)cert_u + Q_{admin}
\end{aligned}$$

10.4 Appendix 4. Source Code

10.4.1 Smart Contracts

```

pragma solidity 0.6.0;
pragma experimental ABIEncoderV2;
contract recordContract
{
    uint recordCount;
    address[] addresses;
    uint[] ids;
    string[] PreviousRecords;
    string[] pkey;
    mapping(address => string[]) private kLookup;
    mapping(address => string[]) private records;
    mapping(uint => address) private userAddress;
    mapping(address=> uint) private countID;
    mapping(address => string) private userType;

    function getPreviousLookup(address userAddress) public view returns(string[]
memory){
        return kLookup[userAddress];
    }

    function addRecord(string memory data, string memory keys) public
    {

        PreviousRecords = records[msg.sender];
        pkey = kLookup[msg.sender];

        PreviousRecords.push(data);
        pkey.push(keys);

        countID[msg.sender] = countID[msg.sender] +1;

        records[msg.sender] = PreviousRecords;
        kLookup[msg.sender] = pkey;

    }

    mapping(address => string) private users;
    mapping(uint=> address) private getAddressFromName;
    mapping(address=> string) private getUserFromAddress;
    mapping(address => mapping(uint => record)) private userRecord;
    mapping(uint=> string) private lookup;

    mapping(address=> uint) private recordLookup;

```

```

struct User {
    string firstname;
    string lastname;
    address userAddress;
}

struct record {
    string id;
    string name;
}

function createUser(address add, uint id, string memory name) public {
    users[add] = name;
    getAddressFromName[id] = msg.sender;
    ids.push(id);
    addresses.push(msg.sender);
}

function createRecord(address add, string memory data, string memory
keys) public {
    PreviousRecords = getPreviousRecords(add);
    PreviousRecords.push(data);
    recordCount = getRecordCount(msg.sender);
    lookup[recordCount] = keys;
    records[add] = PreviousRecords;
}

function getAddressByUser(uint id) external view returns(address){
    return getAddressFromName[id];
}

function getUserByAddress(address userAddress) public view returns(string
memory){
    return users[userAddress];
}

function get() public view returns(string memory){
    return users[msg.sender];
}

function getPreviousRecords(address userAddress) public view returns(string[]
memory){
    return records[userAddress];
}

function getRecordCount(address userAddress) public view returns(uint){
    string[] memory temp = getPreviousRecords(userAddress);

```

```

        return temp.length;
    }

}

pragma solidity 0.6.0;
pragma experimental ABIEncoderV2;

contract fileContract{

    mapping(address => mapping(uint => record)) public userRecord;
    mapping(address => uint) recordCount;
    mapping(address => uint) private countID;
    mapping(address => mapping(uint => string)) public data;
    struct record{

        string recordType;
        string information;
        string date;
        string doctorAddress;

    }

    function uploadRecord(string memory recordType, string memory
information, string memory date, string memory doctorAddress) public{

        userRecord[msg.sender][recordCount[msg.sender]] = record(recordType,
information, date, doctorAddress);
        recordCount[msg.sender] = recordCount[msg.sender] + 1;
    }

    function getRecord(uint recordID) public returns(string memory) {

        return userRecord[msg.sender][recordID].information;
    }
}

contract adminUserContract{

    uint256 public peopleCount;
    mapping(uint => Person) public people;
    mapping(uint=> address) private getAddressFromName;
    mapping(address=> string) private getUserFromAddress;
    address owner;
    address[] addresses;
    modifier onlyOwner(){
        require(msg.sender == owner);
        _;
    }
}

```

```

    struct Person{
        uint id;
        string firstname;
        string lastname;
    }

    constructor() public {
        owner = msg.sender;
    }

    function createUser(string memory firstname, string memory lastname)
    public onlyOwner{

        incrementCount();
        people[peopleCount] = Person(peopleCount, firstname, lastname);
    }

    function incrementCount() internal {
        peopleCount += 1;
    }

    function getAddressByUser(uint id) external view returns(address){
        return getAddressFromName[id];
    }

    function getUserByAddress(address userAddress) private view returns(string
    memory){
        return getUserFromAddress[userAddress];
    }

    function checkAddress(address id) private view returns(bool)
    {
        for(uint i = 0; i < addresses.length; i++)
        {
            if(addresses[i] == id)
            {
                return true;
            }

            else
            {
                return false;
            }
        }
    }
}

```

```

contract doctorContract is fileContract{

    accPatientContract pc;
    fileContract fc;

    constructor() fileContract() public{

```

```

        fileContract fc;
    }

    function checkRecords(uint id) external returns(string memory)
    {
        return fc.getRecord(id);
    }

    function addRecord(string memory recordType, string memory information,
string memory date, string memory doctorAddress ) public
    {

        fc.uploadRecord(recordType,information,date,doctorAddress);

    }
}

contract accPatientContract{
    fileContract patientFc;
}

contract certContract{
mapping(uint=> address) private publicKeys;

    function getAddress(uint id) external view returns(address)
    {
        return publicKeys[id];
    }

    function setAddress(uint id) public
    {
        publicKeys[id] = msg.sender;
    }
}

pragma solidity 0.6.0;
pragma experimental ABIEncoderV2;
contract testPatientContract{

    uint recordCount;
    mapping(uint => User) private users;
    mapping(uint=> address) private getAddressFromName;
    mapping(address=> string) private getUserFromAddress;

```

```

mapping(address => mapping(uint => record)) private userRecord;
mapping(address=> uint) private countID;
mapping(address => record2) public buserRecord;

mapping(address => address[]) public allowedAdd;
mapping(address => mapping(uint => address)) public testAdd;
mapping(address => uint) allowedCount;
address[] getAddressArr;

mapping (address => string[]) public help;
string[] helpBuffer;

    struct User {

        string firstname;
        string lastname;
        address userAddress;
    }

    struct record{

        string recordType;
        string information;
        string date;
        string doctorAddress;
        uint count;
    }

    struct record2{

        string recordType;
        string information;
    }

    string test;
    address[] addresses;
    uint[] ids;
    address[] addressToUser;
    bool[] userExists;

function allowAccess(uint id) public{
    address add = getAddressFromName[id];
    testAdd[msg.sender][allowedCount[msg.sender]] = add;
    allowedCount[msg.sender] = (allowedCount[msg.sender] + 1);
}

function allowed(address patAdd, address docAdd) public view
returns(bool) {

    for(uint i = 0; i < allowedCount[patAdd]; i++ )
    {

```

```

        if(docAdd == testAdd[patAdd][i])
        {
            return true;
        }
    }
    return false;
}

function createUser(uint id, string memory firstname, string memory
lastname) public {

    users[id] = User(firstname, lastname, msg.sender);
    getAddressFromName[id] = msg.sender;
    getUserFromAddress[msg.sender] = firstname;
    ids.push(id);
    addresses.push(msg.sender);

}

function dataSetter(string memory recordType, string memory information)
public view returns(string memory) {

    return string (abi.encodePacked("Record Type: ", recordType , "\n" , "
information: " , information , "\n"));

}

function dataCreate(string memory recordType, string memory information)
public {
    string memory helpmepls = dataSetter(recordType,information);
    helpBuffer = help[msg.sender];
    helpBuffer.push(helpmepls);
    help[msg.sender] = helpBuffer;

}

function getdata() view public returns(string[] memory) {
    return help[msg.sender];
}

function addRecord(string memory recordType, string memory information,
string memory date, string memory doctorAddress ) public
{

    recordCount = userRecord[msg.sender][recordCount].count + 1;
    userRecord[msg.sender][recordCount] = record(recordType, information,
date, doctorAddress,recordCount);
    countID[msg.sender] = recordCount;
    buserRecord[msg.sender] = record2(recordType, information);
}

```

```

function checkRecords(uint id) external view returns(string memory) {

    return string (abi.encodePacked(userRecord[msg.sender][id].recordType , " "
    ,userRecord[msg.sender][id].information, " ",
    userRecord[msg.sender][id].date, " ",
    userRecord[msg.sender][id].doctorAddress));
}

function checkRecords2(uint id, address add) external view returns(string
memory) {

    return string (abi.encodePacked(userRecord[add][id].recordType , " "
    ,userRecord[add][id].information, " ", userRecord[add][id].date, " ",
    userRecord[add][id].doctorAddress));
}

function checkId(uint id) private view returns(bool)
{
    for(uint i = 0; i < ids.length; i++)
    {
        if(ids[i] == id)
        {
            return true;
        }

    }

    else
    {
        return false;
    }
}

function checkAddress(address id) private view returns(bool)
{
    for(uint i = 0; i < addresses.length; i++)
    {
        if(addresses[i] == id)
        {
            return true;
        }

    }

    else
    {
        return false;
    }
}

function getAddressByUser(uint id) external view returns(address){
    return getAddressFromName[id];
}

function getUserByAddress(address userAddress) private view returns(string
memory){

```



```

        return getUserFromAddress[userAddress];
    }

function checkUserExists(uint id, address userAdd) private view
returns(string memory){
    if((checkId(id) == false) && (checkAddress(userAdd) == false))
    {
        return "User Exists";
    }

    else
    {
        return "User Does Not Exist";
    }
}
}

```

10.4.2 Backend Code

```

// Setting up the needed libraries for use
const IPFS = require('ipfs-mini'); // Needed to upload to IPFS
const eciesjs = require('eciesjs'); // Needed for encryption
const ethers = require('ethers'); // Both needed for generating blockchain
addresses
const crypto = require('crypto');
var casual = require('casual'); // Needed to generate random data for use

function generateAddresses() // Using crypto to generate addresses
{
    var id = crypto.randomBytes(32).toString('hex');
    var privateKey = "0x"+id;
    var wallet = new ethers.Wallet(privateKey);
    console.log(wallet.address);
}

generateNames()

function generateNames()
{
    for(var i = 1; i < 10; i++)
    {

```

```

console.log(casual.full_name); // to get a random fullname

}
}

function generateData(){
for(var i = 1; i < 1001; i++)
{
    console.log(casual.words(n = 14)) // generate 14 randoms words for use as
data

}
}

generateData()

function encrypt(passeddata) { // Use ecies to encrypt data

    var k1 = new eciesjs.PrivateKey()
    const data = Buffer.from(passeddata);
    const eData = eciesjs.encrypt(k1.publicKey.toHex(), data)
    console.log(eData)
    return {
        eData : eData,
        k1 : k1
    }
}

function decrypt(data, key){ // Use ecies tp decrypt data

    return dData = eciesjs.decrypt(key, data).toString()
}

//exports funtions for use in browser
exports.encrypt = encrypt;
exports.decrypt = decrypt;

```

10.4.3 Frontend Code

```

<!DOCTYPE html>
<html>
<head>
<link
    href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css"
    rel="stylesheet"
/>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/web3/1.6.1/web3.min.js"
    integrity="sha512-
5erpERW8MxcHDF7Xea9eBQPiRtxbse70pFcaHJuOhdEBQeAxGQjUwgJbuBDWve+xP/u5IoJbKjyJk
50qCnMD7A=="

```

```

        crossorigin="anonymous"
        referrerpolicy="no-referrer"
    ></script>

    <script>src="https://www.jsdelivr.com/package/npm/ipfs-http-
client"</script>
</head>

<body class="h-full">
    <div
        class="
            flex
            w-full
            h-full
            justify-center
            content-center
            items-center
            space-x-4
        "
    >

    <div class="flex flex-col space-y-6">
        <h3 class="text-center">Dashboard</h3>
        <div class="flex flex-col space-y-2">
            <button
                id="loginButton"
                onclick="loginWithEth(), test()"
                class="
                    rounded
                    bg-white
                    border border-gray-400
                    hover:bg-gray-100
                    py-2
                    px-4
                    text-gray-600
                    hover:text-gray-700
                "
            >
                Login
            </button>
            <p id="userAddress" class="text-gray-600"></p>
            <p id="name" class="text-gray-600"></p>
            <p id="demo" class="text-gray-600"></p>
            <button
                id="logoutButton"
                onclick="logout()"
                class="hidden text-blue-500 underline"
            >
                Logout
            </button>
        </div>
        <button
            id="getContractInfo"
            onclick="test3()"
            class="rounded bg-blue-500 hover:bg-blue-700 py-2 px-4 text-white"
        >
            Test

```

```

    </button>
    <button
      id="test2"
      onclick="addAllData()"
      class="rounded bg-blue-500 hover:bg-blue-700 py-2 px-4 text-white"
    >
      test 2
    </button>
  <div id="output"></div>

  <label for="data">Please enter data here: </label>
  <input
    type="text"
    name="data"
    id="data"
    style="border:1px; border-style:solid; border-color:#000000;"
  >

  <button
    id="addData"
    class="rounded bg-blue-500 hover:bg-blue-700 py-2 px-4 text-white"
    onclick="addRecord(document.getElementById('data').value )">Add Data

  </button>
  <div class="flex flex-col space-y-2"></div>
</div>
<script>

//Variable init
const recordArr = [];
const lookupArr = [];
window.userAddress = null;
window.name = null;
var k1;
const CONTRACT_ADDRESS = "0x74322fEa939EBDf12d06506182791661e888ae45";

//Testing Data

//Removed from word document due to its size
function addAllData(){
  for(let i = 0; i < dataArray.length; i++)
  {
    setTimeout(function timer() {
      var address = addressesArr[userNumber[i] -1 ]
      var data = browserify_hello_world.encrypt(dataArray[i])
      var dataToParse = data.eData.toJSON().data

      ipfs.add(dataToParse, (err, result) => {
        if (err) {
          el('#addResponse').innerHTML = 'Hmm.. there was an error: ' +
String(err);
        } else {
          el('#addResponse').innerHTML = result;

```

```

    }
  });

  var result = el('#addResponse').innerHTML

  console.log(address)
  addRecord(result, data.k1, address)
}, i * 5500);

}

}

async function addRecord(data, key, address) {
  const contract = new
window.web3.eth.Contract(window.ABI, CONTRACT_ADDRESS);
  var encryptedData = await contract.methods
    .addRecord(data, key)
    .send({ from: window.userAddress});
}

window.onload = async () => {
  // Init Web3 connected to ETH network
  if (window.ethereum) {
    window.web3 = new Web3(window.ethereum);
  } else {
    alert("No ETH browser extension detected.");
  }

  // Load in Localstore key
  window.userAddress = window.localStorage.getItem("userAddress");
  window.name = window.localStorage.getItem("name");

  showAddress();
};

// Use this function to turn a 42 character ETH address
// into an address like 0x345...12345
function truncateAddress(address) {
  if (!address) {
    return "";
  }
  return `${address.substr(0, 5)}...${address.substr(
    address.length - 5,
    address.length
  )}`;
}

```

```

// Display or remove the users know address on the frontend
function showAddress() {
  if (!window.userAddress) {
    document.getElementById("userAddress").innerText = "";
    document.getElementById("name").innerText = "";
    document.getElementById("logoutButton").classList.add("hidden");
    return false;
  }

  document.getElementById(
    "userAddress"
  ).innerText = `Address: ${window.userAddress}`;
  document.getElementById("logoutButton").classList.remove("hidden");

  document.getElementById(
    "name"
  ).innerText = `Name: ${window.name}`;
  document.getElementById("logoutButton").classList.remove("hidden");
}

// remove stored user address and reset frontend
function logout() {
  window.userAddress = null;
  window.localStorage.removeItem("userAddress");
  window.name = null;
  window.localStorage.removeItem("name");
  document.getElementById("demo").innerHTML = "";
  showAddress();
  document.getElementById('loginButton').style.visibility = 'visible';
}

// Login with Web3 via Metamasks window.ethereum library
async function loginWithEth() {
  document.getElementById('loginButton').style.visibility = 'hidden';
  if (window.web3) {
    try {
      // We use this since ethereum.enable() is deprecated. This method
is not // available in Web3JS - so we call it directly from metamasks'
library
      const selectedAccount = await window.ethereum
        .request({
          method: "eth_requestAccounts",
        })
        .then((accounts) => accounts[0])
        .catch(() => {
          throw Error("No account selected!");
        });

      window.userAddress = selectedAccount;
      window.localStorage.setItem("userAddress", selectedAccount);

      const contract = new window.web3.eth.Contract(
        window.ABI,
        CONTRACT_ADDRESS
      );
    }
  }
}

```

```

    );
    const symbol = await contract.methods
      .getUserByAddress(userAddress)
      .call();

    window.name = symbol;
    window.localStorage.setItem("name", symbol);
    window.records = symbol;
    showAddress();
  } catch (error) {
    console.error(error);
  }
} else {
  alert("No ETH browser extension detected.");
}
}

async function test() {
  let text = "";
  let k = "";
  let d = "";
  const contract = new window.web3.eth.Contract(
    window.ABI, CONTRACT_ADDRESS);
  const count = await contract.methods
    .getPreviousRecords("0x492eDa5B996438A4973E2D9d51C504058f7a0723")
    .call();

  for (let i = 1; i < count.length + 1; i++) {
    text += "Record " + i + ": " + count[i - 1] + "<br>";
  }

  document.getElementById("demo").innerHTML = text;

}

async function getRecords(){

  var text = "";
  var i = 0;
  var x = 0;

  const contract = new window.web3.eth.Contract(
    window.ABI, CONTRACT_ADDRESS);

  const records = await contract.methods
    .getPreviousRecords("0x637dF48d0D8DA04B9647Ff232dB799c60093094a")

```

```

        .call();

const lookup = await contract.methods
    .getPreviousLookup("0x637dF48d0D8DA04B9647Ff232dB799c60093094a")
    .call();

const count = await contract.methods
    .getRecordCount("0x637dF48d0D8DA04B9647Ff232dB799c60093094a")
    .call();

for (let i = 0; i < count; i++) {
    recordArr.push(records[i])
    lookupArr.push(lookup[i])
}

for (let x = 0; x < count ; x++) {

    ipfs.cat(recordArr[x], (err, result) => {
        if (err) {
            el('#lookupResponse').innerHTML = 'Hmm.. hmm there was an error:
' + String(err);
        } else {
            var parse = result.split(',').map(function(item) {
                return parseInt(item, 10);
            });

            var bb = browserify_hello_world.dec(Uint8Array.from(parse),
lookupArr[x])
            text += "Record " + x + ": " + bb + "<br>";
            document.getElementById("demo").innerHTML = text;

        }
    });

}

}

function dtest(pData) {
    d = pData

}

function getdtest() {
    var result = d
    return result

}

```



```

    async function getContractSymbol() {

        const contract = new window.web3.eth.Contract(
            window.ABI, CONTRACT_ADDRESS);
        const symbol = await contract.methods
            .getUserByAddress(userAddress)
            .call();

    }

</script>
<script src="out2.js"></script>
<h4>Add Data</h4>
<textarea id="addInputData" placeholder="Hello world!"></textarea>
<button id="addData2">Add Data</button>
<input type="text" id="lookupHash"
placeholder="QmbhrsdbvQy3RyNiDdStgF4YRVc4arteS3wL5ES5M6cVd" />
<button id="lookup">Lookup Block</button>
<button id="lookupStats">Get Stats</button>
<div id="addResponse"></div>
<div id="k1"></div>
<div id="lookupResponse"></div>
<script type="text/javascript" src="../../dist/ipfs-mini.js"></script>
<script type="text/javascript">
    var el = function(id){ return document.querySelector(id); };
    var ipfs = new IPFS({ host: 'ipfs.infura.io', protocol: 'https' });

    el('#addData2').addEventListener('click', function(){
        var isJSON = true;
        var inputData = el('#addInputData').value;
        var eData = browserify_hello_world.encrypt(inputData)
        var dataToParse = eData.eData.toJSON().data
        var bb = JSON.stringify(eData.k1)
        var xx = eData.k1.toHex()

        ipfs.add(dataToParse, (err, result) => {
            if (err) {
                el('#addResponse').innerHTML = 'Hmm.. there was an error: ' +
String(err);
            } else {
                el('#addResponse').innerHTML = result;

                dtest(xx)
                addRecord(result, xx)
            }
        })
    })

```

```

    }
  });

});

el('#lookup').addEventListener('click', function(){
  var isJSON = true;
  var lookupHash = el('#lookupHash').value;
  ipfs.cat(lookupHash, (err, result) => {
    if (err) {
      el('#lookupResponse').innerHTML = 'Hmm.. hmm there was an error:
' + String(err);
    } else {
      getRecords()
      el('#lookupResponse').innerHTML = result;

      var parse = result.split(',').map(function(item) {
        return parseInt(item, 10);
      });

      var c = getdtest();
      var bb = browserify_hello_world.dec(Uint8Array.from(parse), c)

    }
  });

});

});

</script>
</body>
</html>
<script>
window.ABI = [
  {
    "inputs": [
      {
        "internalType": "string",
        "name": "data",
        "type": "string"
      },

```

```

        {
            "internalType": "string",
            "name": "keys",
            "type": "string"
        }
    ],
    "name": "addRecord",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "address",
            "name": "add",
            "type": "address"
        },
        {
            "internalType": "string",
            "name": "data",
            "type": "string"
        },
        {
            "internalType": "string",
            "name": "keys",
            "type": "string"
        }
    ],
    "name": "createRecord",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "address",
            "name": "add",
            "type": "address"
        },
        {
            "internalType": "uint256",
            "name": "id",
            "type": "uint256"
        },
        {
            "internalType": "string",
            "name": "name",
            "type": "string"
        }
    ],
    "name": "createUser",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
},

```

```

{
  "inputs": [],
  "name": "get",
  "outputs": [
    {
      "internalType": "string",
      "name": "",
      "type": "string"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "uint256",
      "name": "id",
      "type": "uint256"
    }
  ],
  "name": "getAddressByUser",
  "outputs": [
    {
      "internalType": "address",
      "name": "",
      "type": "address"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "address",
      "name": "userAddress",
      "type": "address"
    }
  ],
  "name": "getPreviousLookup",
  "outputs": [
    {
      "internalType": "string[]",
      "name": "",
      "type": "string[]"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "address",
      "name": "userAddress",
      "type": "address"
    }
  ]
}

```

```

    }
  ],
  "name": "getPreviousRecords",
  "outputs": [
    {
      "internalType": "string[]",
      "name": "",
      "type": "string[]"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "address",
      "name": "userAddress",
      "type": "address"
    }
  ],
  "name": "getRecordCount",
  "outputs": [
    {
      "internalType": "uint256",
      "name": "",
      "type": "uint256"
    }
  ],
  "stateMutability": "view",
  "type": "function"
},
{
  "inputs": [
    {
      "internalType": "address",
      "name": "userAddress",
      "type": "address"
    }
  ],
  "name": "getUserByAddress",
  "outputs": [
    {
      "internalType": "string",
      "name": "",
      "type": "string"
    }
  ],
  "stateMutability": "view",
  "type": "function"
}
];
</script>

```