electronics MDPI

Article

Towards a Blockchain Assisted Patient Owned System for Electronic Health Records

Tomilayo Fatokun [1], Avishek Nag [2,*] and Sachin Sharma [3,*]

[1] School of Computing, National College of Ireland, Dublin 1, Ireland; x19187882@student.ncirl.ie
[2] School of Electrical and Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland
[3] School of Electrical and Electronic Engineering, Technological University Dublin, Grangegorman, Dublin 7, Ireland
* Correspondence: Avishek.Nag@UCD.ie (A.N.); Sachin.Sharma@TUDublin.ie (S.S.)

Abstract: Security and privacy of patients' data is a major concern in the healthcare industry. In this paper, we propose a system that activates robust security and privacy of patients' medical records as well as enables interoperability and data exchange between the different healthcare providers. The work proposes the shift from patient's electronic health records being managed and controlled by the healthcare industry to a patient-centric application where patients are in control of their data. The aim of this research is to build an Electronic Healthcare Record (EHR) system that is layered on the Ethereum blockchain platform and smart contract in order to eliminate the need for third-party systems. With this system, the healthcare provider can search for patient's data and request the patients' consent to access it. Patients manage their data which enables an expedited data exchange across EHR systems. Each patient's data are stored on the peer-to-peer node ledger. The proposed patient-centric EHR platform is cross-platform compliant, as it can be accessed via personal computers and mobile devices and facilitates interoperability across healthcare providers as patients' medical records are gathered from different healthcare providers and stored in a unified format. The proposed framework is tested on a private Ethereum network using Ganache. The results show the effectiveness of the system with respect to security, privacy, performance and interoperability.

Keywords: blockchain; healthcare; ethereum platform

check for updates

Citation: Tomilayo, T.; Nag, A.; Sharma, S. Towards a Blockchain Assisted Patient Owned System for Electronic Health Records. *Electronics* **2021**, *10*, 580. https://doi.org/10.3390/electronics10050580

Academic Editor: Mehdi Sookhak

Received: 5 February 2021
Accepted: 24 February 2021
Published: 2 March 2021

1. Introduction

According to the Fierce healthcare report (https://www.fiercehealthcare.com/tech/number-patient-records-breached-2019-almost-tripled-from-2018-as-healthcare-faces-new-threats (accessed on 5 Feburary 2021)), patients health data breach in the healthcare industry tripled in the year 2019 compared to past years as a result of little to non-existent security measures in Electronic Health Record (EHR) systems. Further, because of remote working due to COVID-19 restrictions, the impact of these breaches is far-reaching and includes the cases where patients' private data are sold online for profit, and patients are being blackmailed with threats to expose their data in public. Other issues with traditional EHR systems include control and hoarding of patients' health data by health providers which may lead to delays in data exchange between EHR systems which may ultimately lead to delays in providing timely healthcare to the patients [1]. There is also the problem of interoperability between different EHR systems. Due to the above challenges, there is a need to develop an EHR system with advanced security at its core as well as decentralize the control of patient data.

Recently, European Commission has adopted a recommendation on a European EHR exchange format [2]. According to this recommendation (https://ec.europa.eu/digital-single-market/en/news/recommendation-european-electronic-health-record-exchange-format (accessed on 5 Feburary 2021)), citizens of the EU should be able to access and exchange their electronic health (e-health) data securely with any healthcare expert whenever they are in the EU. Further, in [3], Samarin also discussed the need for the transfer of full ownership of medical records to patients. In [4], the effect of providing patients online access to e-health data is investigated. It has been concluded that providing online access to their health data offers increased convenience and satisfaction. However, there are concerns related to the workload, security and privacy.

In this paper, we propose a system that has the capability to meet the security, privacy, interoperability and performance requirements of an EHR system. Using the proposed system, patient data can be shared anytime and anywhere with the permission of the patient. The existing work such as [5–8] deals with a few of the above requirements in designing such an EHR system. However, in this work, we consider all the above requirements so that security and privacy of patient data are maintained, interoperability is achieved, and performance can be met. To meet these requirements, this work takes the advantage of blockchain technology to leverage its advanced and robust cryptography mechanism and enables cross-healthcare-provider data exchange while enabling the patient with control over their data. The purpose of the study is to design such a blockchain-based EHR application and test it for security, privacy, interoperability and performance requirements.

Blockchain is a secure decentralized network node that uses the peer-to-peer network to exchange data. The benefit of blockchain includes transparency, security, immutability, audibility. Currently, there exist several blockchain platforms such as Ethereum (https://ethereum.org/ (accessed on 5 Feburary 2021)), QTUM (https://qtum.org/ (accessed on 5 Feburary 2021)), NEO (https://neo.org/ (accessed on 5 Feburary 2021)) and Cardano (https://cardano.org/ (accessed on 5 Feburary 2021)). The Ethereum platform is a leading platform to implement smart contracts and blockchain-based applications and is globally accepted as an advanced blockchain platform to perform various tasks (e.g., security exchange data) that could be used by various industries, not limited to the financial industry. Therefore, we use this platform in our proposed blockchain-based framework. Using this platform, the patient-centric framework proposed in this work uses blockchain smart contracts to store patients' data and executes functions in a decentralized system. Once the smart contract is deployed, transactions are sent through it including security and privacy features. Further, requested changes made by the transactions can be mined and broadcasted to the entire decentralized systems.

In our framework, we also use MetaMask (https://metamask.io/ (accessed on 5 Feburary 2021)) to store the keys of patients and healthcare providers. MetaMask is a cryptocurrency wallet that can be used on different browsers such as Chrome, Firefox and Edge browsers. There exist several other crypto wallets such as MyEtherWallet (https://www.myetherwallet.com/ (accessed on 5 Feburary 2021)) and MyCrypto (https://mycrypto.com/ (accessed on 5 Feburary 2021)). The aim of MetaMask is to create a simple, reliable and secure way to connect with the Ethereum platform. Further, since the creation of MetaMask in 2016, it has not faced major hacks. Therefore, we use MetaMask in our work. In the proposed system, MetaMask serves as the bridge between normal browsers and the Ethereum blockchain. MetaMask stores the keys for Ethereum and ensures that only the healthcare providers and patients with the private keys can access the application.

The contributions of the paper are as follows:

1. Design and implementation of a patient-centric EHR web portal front-end platform.
2. Integrating the above patient-centric EHR into the Ethereum blockchain platform and its smart contract.
3. Ensuring that a patient's health record is secure, consistent, and available across healthcare providers and decided upon by the patient.
4. Perform experiments on security, privacy and interoperability of the proposed blockchain-enabled framework.

Using our proposed framework, the healthcare providers can search for patient's data and request the patient consents to access it. Patients manage their data which enables an expedited data exchange across EHR systems. Each patient data is stored on the peer-to-peer node ledger. The proposed framework is tested on a private Ethereum network using Ganache. The results show the effectiveness of the system with respect to security, privacy, performance and interoperability.

Section 2 provides the background information and the related work, Section 3 presents the detail about our proposed framework, Section 4 provides implementation details and results and finally, Section 5 concludes.

2. State-of-the-Art

This section first provides the background of blockchain and Ethereum blockchain and then provides the related work.

2.1. Blockchain

Blockchain is a distributed ledger technology that enables users to interact, store and retrieve data with ensured data authenticity, immutability, and non-repudiation. This eliminates the need of centralized system, due to distributed nature of blockchain that enables the entities and various devices to exchange data, to and from their peers. The first block in a blockchain is referred to as the genesis block, which does not contain any transaction. Each block thereafter contains a number of validated transactions and is cryptographically linked with the previous block as shown in Figure 1. Blockchain provides a robust security structure by utilizing cryptography mechanisms for encryption.

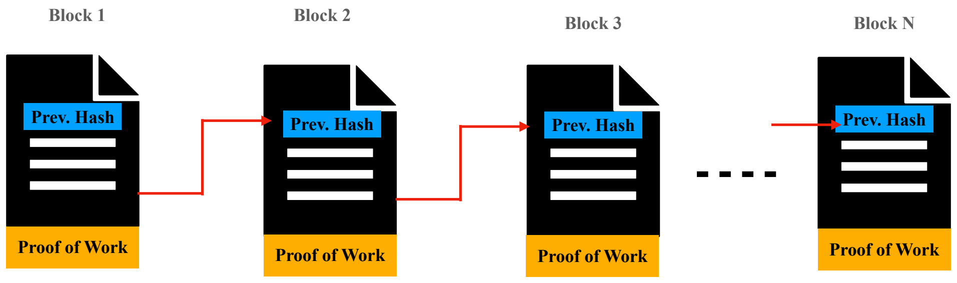

Figure 1. Blockchain schematics.

2.2. Ethereum Blockchain

Ethereum is a distributed blockchain network with the idea to create a trustless smart contract platform that is open-source and has programmable blockchain features. It functions as a global network of computers that work together as a supercomputer to create, manage, and run decentralized digital applications referred to as "dapps". This enables users to make agreements and conduct transactions directly with each other to buy, sell and trade goods and services without a middle man (trustless smart contracts). With trustless smart contracts, middlemen like the banks can be bypassed for financial transactions, legal contracts can be drawn up and executed without the need of lawyers, and users can launch their own e-commerce sites without going through a large corporation. The Ethereum platform also owns its own cryptocurrency known as Ethers [9]. This cryptocurrency can be used for sharing between accounts connected on the Ethereum blockchain [10]. Ether is the standard payment unit in the Ethereum platform. Ethereum also provides the programmers a language in which they can customize their own blockchain. This language is known as Solidity. It was created for use in developing smart contracts. Ethereum makes use of Elliptic Curve Cryptography (ECC) and Elliptic Curve Digital Signature Algorithm (ECDSA) for its public-key cryptography.

2.3. Related Work

Table 1 provides the summary of the related work. The first column presents the reference number, the second column provides the technologies used in the related work and the third column provides the information about the features considered. In this work, we are interested in the following four features: (1) interoperability, (2) security, (3) response time (i.e., performance) and (4) compatibility. The descriptions of each of the related work are given below:

Table 1. Existing work. I means interoperability, S means security, R means response time, C means compatibility, * means database and ** means authentication, ✓ means the feature is explored and × means the feature is not explored.

Reference	Technologies Used	Feature Explored (✓) or not (×)
[3]	Cloud, and deposit box	I (✓), S (×), R(×), C(×)
[6]	Service Oriented Architecture	I (✓), S (✓), R (×), C (×)
[5]	Archetype-Based Solution	I (✓), S (✓), R (×), C (✓)
[11]	Cloud-Based Solution	I (✓), S (✓), R (×), C (×)
[7]	Cloud and Web Service API	I (✓), S (✓), R (×), C (✓)
[8]	Peer to Peer Communication	I (✓), S (×), R (✓), C (✓)
[12]	Blockchain	I (✓), S * (×), R (✓), C (✓)
[13]	Blockchain	I (✓), S ** (×), R (✓), C (✓)
[10,14]	Ethereum blockchain	I (✓), S (✓), R (✓), C (✓)

In [3], medical records that are privately stored on the cloud and solely accessible by the patient are discussed. This evidently transfers full ownership of medical records to patients. However, it neglects the need to share medical information with multiple entities such as health providers. Further, this work addresses the issue of interoperability by employing the use of a deposit box when a record is transferred to another party. However, the concept proposed in this work does not address the situation in which a doctor has to keep the patient's medical records undisclosed even from the patient. Moreover, this work does not address the security aspect of medical records. In [6], the Service Oriented Architecture (SOA) was proposed to support the reuse and exchange of diverse patient records across units of the same organization and beyond. This approach was also used for the integration of the Electronic Health Records (EHR) systems and Personal Health Record (PHR) systems to solve interoperability, reusability and security problems of PHR systems. The challenge with this approach is the compatibility with other PHR systems and patient's data privacy and security were not fully addressed.

The interoperability problem faced by Clinical Decision-Support Systems (CDSSs), and electronic health records systems is discussed in detail at [5]. In this work, a solution based on the archetype is proposed The archetype is defined as the data standard that needs to be exchanged between two systems using schema patterns. One of the advantages of this system is feeding the EHR with data derived from the mechanism of abstraction. The problem is that the issue with system efficiency response time was not addressed in this work. Further, in [11], the design is proposed for a secure, interoperable, and cloud-based PHR service. The Continuity of Care Document (CCD) was used to optimize portability and interoperability during the storage and exchange of the PHR data of a patient. A broad spectrum of security mechanisms that includes access control, encryption, and digital signature in an integrated, embedded, and fine-grained manner, based on open standards such as Extensible Access Control Mark-up Language (XML), XML Encryption, XML Signature, and XML Key Management Specification were used for providing self-protecting security for each CCD instance.

In [7], a cloud-based application and web-service Application Programming Interface (API) to control and access shared data is proposed. The proposed framework was tested using a comparative usage scenario received from community care. The system is controlled by the Ministry of Health to manage the hospital or health centers with an Electronic Health Record (EHR) system. The aim of the proposed system is to increase interoperability of healthcare data with different healthcare record systems and healthcare providers taking into account these three components of healthcare integration which are minimum dataset, platform-independent technology and the governing body. The challenge with this proposed framework was that the prototype is limited to a limited dataset.

Peer-to-peer communication using JXTA technology was designed to solve electronic health interoperability issues in [8]. Blekinge County healthcare organization was used as a case study. JXTA was used as a suitable Java-based open-source platform for P2P communication as it is compatible with several platforms. The peer was used to represent the healthcare center and each system was connected to the internet. When a system needs patient information, a request is sent. The other system searches the database which is connected to the P2P platform. For the purpose of the case study, the database was shared to jxa.org and once the other was able to retrieve the data from the database, the information was shared with the system. In [12], blockchain is used to store healthcare. With the proposed system, patients have secure access to their medical records. With the permission management feature, patients are fully aware of changes made on their records and this also checks the data type to be displayed to each blockchain miner. This framework used consensus algorithms such as Proof-of-work (POW) to authenticate new blocks created in the blockchain system and smart contract. The system can accept data from different sources. The problem with this system was that the system does not discuss the database security issues.

In [13], a blockchain framework that mediates between the users and the pool of sensitive shared data was proposed. This system was compared with other blockchain platforms such as bitcoin. This system verifies patient's data by employing the use of cryptography schemes. The end-to-end test of this system showed that the used technique is lightweight, scalable and effective. It is further stated that future research work needs to be carried out on authentication and communication protocols. Further, in [15], the use of symmetric key encryption pair and authorized digital signature is used to verify participants through their membership service API (Application Programming Interface). This prototype focuses on cancer-affected patients. This framework helps to reduce turnaround data sharing time and also guarantees security and privacy.

As shown in Table 1, the Ethereum blockchain platform is used in [10,14]. In [14], the proposed system is called MIStore, a medical insurance storage system based on blockchain. This utilizes less memory and CPU. In [10], Ethereum blockchain is used for Electronic Health Record and Electronic Medical Record. Like [10,14], this paper considers security, privacy, interoperability, compatibility and performance based on the Ethereum blockchain, and comes up with the framework which has the capability to achieve all the requirements. The proposed work in this paper is in the direction where only patients own their own data and they will give permission to the third party to process it.

3. Proposed Ethereum Blockchain Assisted Framework

This section first provides the basic introduction to our proposed framework and later discuss the framework in detail.

3.1. Basic Introduction to the Proposed Framework

Figure 2 shows the summary of proposed smart contract framework for Electronic Health Record (EHR). This smart contract contains the definition of terms of the agreement, how the users interact with the Ethereum blockchain. The administrator (Admin in Figure 2) acts as a governing body that is in charge of the registration of healthcare institutions on the application. The administrator is able to create, edit, add and suspend hospital accounts. When a new patient record is created, a new block is created at the back-end, verified and broadcasted to all the nodes on the network. The new block is added to the blockchain. The system is designed to enable the hospitals to view and create patient medical records after an access request has been sent to the patient, and the patient has granted access. The request is done by the hospital in cases where a different doctor has to attend to a patient. A denied request means the hospital and its staff cannot access the patients' healthcare records. The hospital can also schedule an appointment with patients. The patients' terms of interaction is defined by two activities: (1) granting or denying access to healthcare records and (2) viewing appointment logs. This design ensures that data exchange is easier and data security, privacy, and integrity is maintained, as blockchain is used as an enabling technology to share records.

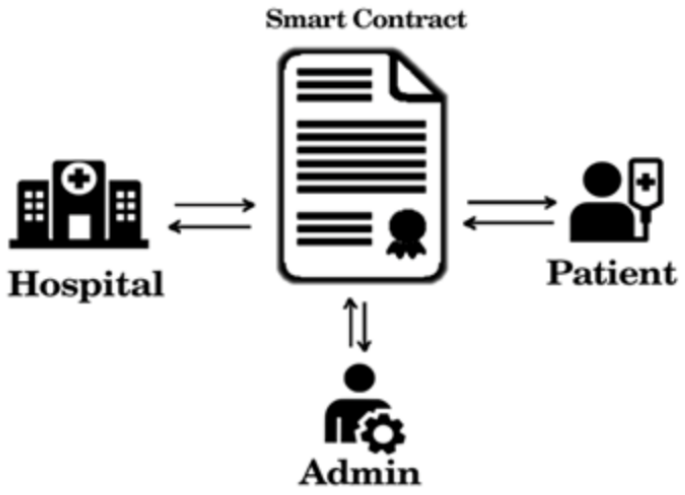

Figure 2. Basic Smart contract Framework for Electronic Healthcare Record System.

3.2. Design Consideration

In this research, we focused our implementation on three important parameters: (1) security of patient data, (2) interoperability between the healthcare providers and (3) performance. We took the advantage of blockchain technology which is highly secured and immutable to store data. To enforce interoperability, patient data from diverse healthcare providers are maintained in a standard format. In this format, a single format is used to store the data from the different providers, so that interoperability can be obtained. Further, performance is increased by adopting the decentralized nature of the blockchain.

The Ethereum blockchain platform uses the keccak-256 cryptographic hash function in its algorithm to generate a hash for data stored in the blockchain. Each transaction is treated as a block and Ethereum digitally signs every data stored inside each block in the blockchain. Every block in the existing blockchain has a timestamp, a unique identifier known as a hash and maintains the unique identifier of the preceding block. In our case, a transaction could be adding a new patient record to the system. Whenever a new patient is added, a new block gets added on the existing blockchain architecture. All transactions within the blockchain are validated through a consensus mechanism where all the nodes in the environment verify and agree on a block before it is added to the chain. Additionally, once a record is added to the chain, arbitrary modifications by participants are not allowed [16]. The blocks are connected with each other and have a distributed and decentralized network. Any tempering on the block once added to the blockchain causes it to be invalidated. This ensures easy detection of malicious access to the network, and enforces the integrity of the data in the chain. Because each block has a distinct signature from the next, any attempt to hack the network becomes improbable and very costly to the hacker. Ethereum also utilizes the consensus algorithm, Proof of Work (PoW) as a fault-tolerant mechanism to ensure the necessary agreement on a single state of its network all the distributed participants in the network.

To solve the issue of interoperability in EHR, some healthcare organizations use Fast Healthcare Interoperability Resources (FHIR), while others use the Clinical Document Architecture (CDA) standard for data exchange or HL7 2.x standard. These varying data standards directly reduce interoperability. In this paper, we use blockchain, which helps in overcoming this challenge by accessing data through APIs. This achieves standardization of data formats enabling data transmission in a single format, irrespective of the capabilities of EHRs.

Our proposed system avoids a single point of failure by decentralizing the system and gives patients the right to ownership and access to their own data. The entire healthcare data is easy to access, secured, and distributed. Any illegal changes made to the blockchain data will be easily detectable and identified. Legitimate consent has been actualized in the proposed framework, which keeps an eye on what kind of information will appear to which blockchain miners. The proposed implementation is motivated by the work by Shahnaz et al. (2019) [10].

3.3. Proposed Blockchain Framework for EHR in Detail

Figure 3 shows the proposed framework in detail. It consists of three major components, namely User Layer, Front End layer, and Back End layer. The user layer refers to the system user classes which are made up of the system administrator, hospital staff and patients modules. These modules are described below:

1. The administrative module implements intuitive and easy-to-use application user interfaces. It has six sub-modules which enables the administrator to perform the tasks managing hospital accounts. Below is a list of all administrator sub-modules
   - Dashboard Module—This module loads and lists all registered hospitals on the consortium blockchain network. The administrator can explore these hospital records and manage the hospital by using the explore link provided by the displayed hospital element on the dashboard.
   - Create-Hospital-Record Module—This module enables the administrator to create or add a hospital record. It is a fully validated web form and holds the programming logic used to register and create hospital records.
   - Hospital-Record-Details Module—This module empowers the administrator to view hospital records and perform other administrative tasks addressing the particular needs of the hospital at the time. This administrative task may include add staff or employees to the hospital, edit hospital record, suspend hospital account, etc.
   - Edit-Hospital-Record Module—This module enables the administrator to make a change to hospital records.
   - Create-Hospital-Staff-Record Module—The administrator uses the module to add hospital staff records to the currently selected hospital records
   - Administrator-Menu Module—The module is embedded into every other module in the administrator module. There are two variants. The first variant is used on the Administrator-Dashboard and Create-Hospital-Record module while the other variant is used with the Hospital-Record-Details and Create-Hospital-Staff-Record Module.
2. Hospital Module
   In the administrative module, the hospital module implements user-friendly and easy-to-use application user interfaces. It has seven sub-modules which enables the logged users to perform the tasks that help the hospital manage their clients. Below is a list of all hospital sub-modules:
   - Request-Access-to-Patient-Medical-Records Module: This module allows logged in hospital staff (Doctors) to request access to the patient medical record. The module implements a search functionality to enable the doctor to retrieve the patient record using the public key or account token. The application uses the provided token to search the patient's records from the back-end and if the patient record exists, the doctor then requests access to the patient record and awaits the approval for the request. In the case where the user account is not found, the doctor is prompted to register the patient and create the patient medical

report, and request access to the patient record. Upon approval of the request made by the doctor, the doctor can now access and view patient medical records.

- Patient-Medical-Record Module: The logged hospital staff granted access to patient medical records the proceed to view the patient details and medical history information. This module also equips the hospital staff with a set of tools to update patient records after due consultation with the patient and to schedule an appointment for a subsequent meeting with the patient. After successful completion of the consultation, when the doctor exits the patient medical record page, the doctor's access authorization is revoked. The doctor will need to request access again if he/she must access the record again.
- Add-Medical-Report Module: This module enables the logged hospital staff to create a medical record based on the consultation and findings during the session the doctor had with the patient and update the patient medical history.
- Dashboard Module: This module equips the logged hospital staff with the stats on how many patients the hospitals have attended for the day, the number of scheduled appointments for the day, and the number of appointments the hospital is expected to keep for that day. This module also lists to the doctor all approved medical record access requests and the list of appointments that have been kept for the day by the hospital.
- Hospital-Appointment Module: This module shows the hospital appointment logs for the logged user use. The appointments for the current day in view are listed first, then all other appoints are listed below the day's appointments.
- Schedule-Appointment Module: This module enables the logged-in hospital staff to create an appointment for the patient's next visit to the hospitals. This module is a fully validated form and when completed with appropriate data generates an appointment record.
- Hospital-Menu Module: The module is embedded into every other module in the hospital module. The module is a list of links giving the logged user access to other resources.

3. Patient Module
   The patient module implements user-friendly and easy-to-use application user interfaces that enable logged-in patient users to interact with the application. It has six sub-modules and these include:
   - Dashboard Module: This module shows the logged patient notifications i.e., alert logged users that new request to access their record by a hospital and or displays the appointment the patient has for the day. This module also displays profile information for the logged-in user.
   - Grant-Access-to-Request Module: This module enables the logged user to access all requests made by hospitals to access their medical records and also provides the mechanism for approving and rejecting the request from the hospitals. When a request is approved by a patient the request is active for that day and the access is revoked when the requesting hospital completes the consultation with the patient. Note the patient can cancel any request initially granted by the patient at any time and that action is validated on the system.
   - Patient-Medical-History Module—This module enables the logged user to view their medical history. The logged user cannot modify or tamper with the information in their medical history, they can only view the records.
   - Patient-Menu Module—The module is embedded into every other module in the patient module. The module is the list of links giving the logged user access to other resources.

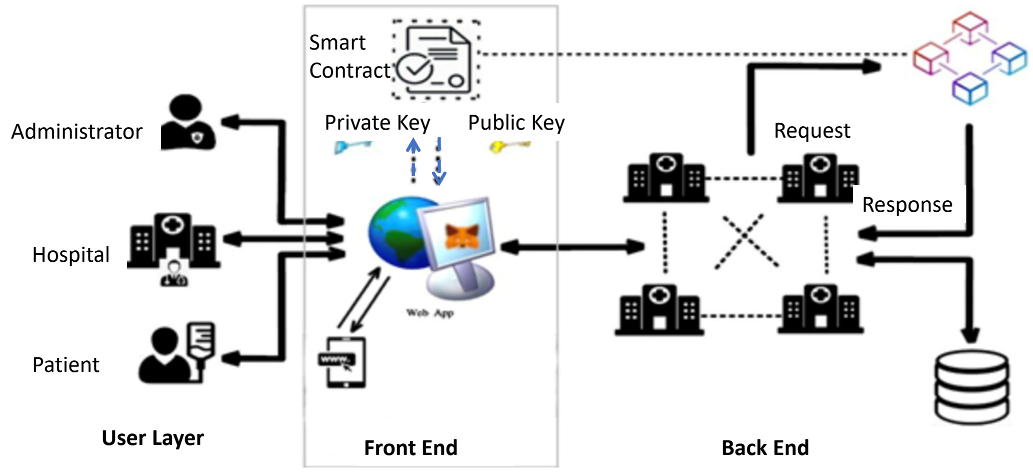

Figure 3. Blockchain framework for Electronic Health Records in detail.

The frontend (shown in Figure 3) is the dapp application that creates the smart contract and the Backend is the smart contract deployed on the Ethereum network. To deploy the smart contract, the smart contract will first compile the smart contract and generate the application binary interface (ABI) which is the json representation of the smart contract in terms of agreement for usage. The compiled smart contract will then be deployed to the Ethereum network using the migration tools. The deployed smart contract will be made accessible for the front-end where MetaMask can interact with the deployed file. The application (dapp) can be accessed using a web browser and MetaMask, a web3 protocol for accessing a decentralized network from your browser. To login to the application the user (Administrator, Hospital Staff, or Patient), the private key for the user will be used to import the user account to the browser via MetaMask and connect to the Ethereum network.

The public key on the Ethereum network is used to address and identifies an account on the network. A request from the dapp application (login request) triggers the sending transaction to the backend (smart contract). The sent transaction attracts a processing fee called gas. The gas on the Ethereum network is the required fee used to successfully conduct a transaction or execute a contract on the platform. Upon the login request initiation and the confirmation of the request on the MetaMask dialog pop up, the transaction is then forwarded to the backend where transaction will process. To process a transaction, the miner uses mining algorithms to mine new blocks on the blockchain. The mined block might consist of multiple transactions and each transaction is verified and validated across the network. Upon the successful mining of the block and transaction, the response from the backend is sent to the frontend and updates the user UI based on the content of the server response. This process is repeated for every transaction that is initiated by the user.

4. Implementation and Results

The proposed application system was implemented by developing two major components: (1) the web application developed which is the patient-centric EHR using JavaScript (Nodejs) and (2) a smart contract that is written in Solidity which is an Ethereum programming language with JavaScript and Python encapsulated in it. This application was developed on an Intel i5 core processor of 8 GB RAM and Windows 64-bit version 10 platform. Ganache (Truffle Suite) was used as Ethereum blockchain and its CLI (command line interface) was used as an Ethereum client that connects to a local blockchain for testing our decentralized application. Further, MetaMask was used in the implementation of our application and to communicate with the Ethereum blockchain. This is a Google Chrome, Vivaldi, Opera, and Firefox extension for the browser which makes it easy for web applications to communicate with the Ethereum blockchain. Visual studio code is a popular text editor developed by Microsoft as an open-source project. The application was written and edited using this text editor. web3.js is a collection of libraries that are used to interact with the Ethereum node, using an Hyper Text Transfer Protocol (HTTP) or Inter Process Communication (IPC) connection.

Table 2 shows the software and associated versions used to implement the proposed framework. The Vue.js UI technology was used to develop the application user interface (UI). Below are some of the criteria for selecting this UI technology:

1. It was a progressive framework that allowed for the quick development of web interfaces with a better user experience.
2. It was a multi-platform UI technology as it supported both web and mobile platforms.
3. It was lightweight and thus guaranteed a higher performance compared to other UI technologies.
4. It had cross-platform support.
5. It was flexible and easy to integrate with an existing application.

Table 2. Software used with versions associated.

Required Software	Version
Solidity	v0.5.0
Truffle Suite	v5.0.2
Nodejs	v14.15.1
Web3js	v8.1.8
Visual Studio Code	v1.51.1

4.1. Evaluations

With the evaluation, we provide details about the tests done on the implemented system to appraise if the outcome met the requirements of the proposed system. Four different types of evaluations were performed: (1) functional testing, (2) performance testing, (3) security testing and (4) interoperability testing. The objective of the functional testing was that each implemented functionality should work as expected. The objective of the performance testing was that the maximum response time of our application was like a web application i.e., less than 1 second, as users would not likely notice a delay [17]. The security testing was performed such that if a user tried to enter a wrong key, he/she should not be able to access the data. The interoperability testing was performed to test that data from the different healthcare providers should be accessed/processed through our platform.

4.1.1. Functional Testing

To successfully deploy a smart contract on the Ethereum blockchain, the generated ABI file which was a JSON representation of the terms defined in the smart contract contained valid information that could be used to test all the components built into the smart contract. To ensure that the smart contract was successfully deployed every time the application was executed, two sets of the test were written. Below are the test details:

1. Deployment Test:
   For the deployment test, we verified that the smart contract has a valid address and the name of the application is set to define the smart contract.
2. Administrator Entity Test:
   For the administrator entity test, four different tests were written to ensure that the administrator entity would be deployed properly and function as anticipated in the production environment. For the administrator entity, three different tests were written to ensure that the administrator entity would be deployed properly and function as anticipated in the production environment. In the first test, it was verified that the default administrator account was created. In the second test, it was verified that attempts to create the new administrator account were successful. In the third test, it was verified that the administrator login worked fine.

3. Hospital Entity Test:
   The smart contract defined a 'create' functionality for the hospital entity and the test written was to ensure that it created hospital record work as expected. A second test was written to retrieve all hospital records in the back-end. Below are the test details. The following were verified in these tests: (1) the hospital account could be created and (2) the hospital accounts could be listed.
4. Hospital Staff Entity Test:
   Three different tests were written to ensure that the smart contract implementation of the Hospital staff records was well defined and it worked as expected. The following were verified in these tests: (1) it was checked that the hospital staff account was created, (2) the list of the hospital staff accounts could be retrieved, (3) the hospital login worked.
5. Patient Entity Test:
   For the patient entity, three different tests were written to ensure that the patient entity would be deployed properly and functioned as anticipated in the production environment. The following were verified: (1) it was checked that a patient account was created, (2) the list of patients could be retrieved and (3) patient login worked fine.
6. Patient Medical Record Entity Test:
   For the Patient Medical Record entity, two different tests were written to ensure that the patient medical record entity would be deployed properly and functioned as anticipated in the production environment. The following were verified: (1) patient medical records could be created and (2) the patients' medical records could be retrieved.
7. Access Patient Medical Record Request Entity Test:
   In these tests, two cases of access to patient medical record were verified: (1) a request for access to patient medical record could be made and (2) the approval or denial of access to the patient medical record was working fine.
8. Scheduled Appointments Entity Test:
   In these tests, two cases were verified: (1) an appointment with a patient could be scheduled and (2) the appointment logs could be viewed.

4.1.2. Smart Contract Execution Evaluations

The application runs on a Dapp (Web browser). For the experiment, we employed integration testing and executed the smart contract. Three hospitals (Hospital A, B and C) and 30 patient records were configured in our experiments. The execution time was calculated and is shown in Figure 4.

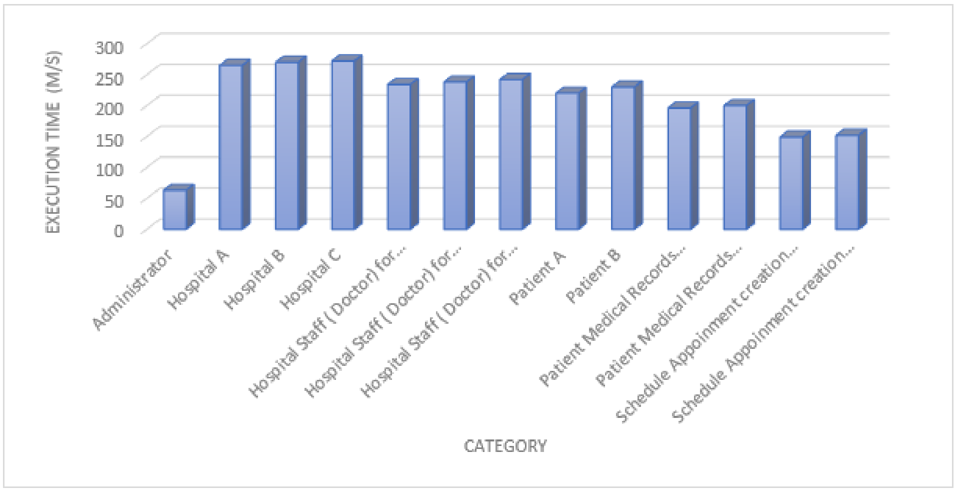

Figure 4. Execution time in milliseconds.

The execution time for different categories of users such as administrator, hospitals (Hospital A, B and C), hospital staff and two patients (patient A and B) are shown in Figure 4. The execution time fluctuated as the number of transactions increased with little difference. These transactions were performed for various events trigger in the smart contract. We had one user (Administrator) using the system functions such as 'Create Administrator', and 'add three Hospitals'; it was 63 milliseconds, 265 milliseconds and 270 milliseconds respectively for these functions to be executed. This time was less than 1 second. Therefore, it might be unnoticeable to a user.

4.1.3. Security Evaluations

A security test was performed and the patient's data was proven to be secured as entities on the user layer with the wrong key could not access the patient-centric EHR application. Only the right key for Ethereum which was a Keccak-256 hash encoded as a hexadecimal string of length 64 could access the application. Each transaction was encrypted with the patient's public key which was based on cryptography and could only be decrypted using that patient's secure private key. Ethereum used Elliptic Curve Digital Signature Algorithm (ECDSA) to provide security and privacy for all the nodes on the network.

4.1.4. Interoperability Evaluations

Interoperability in the healthcare industry has been a serious issue where it is difficult to exchange medical records. In this paper, an interoperability test was carried out as different hospitals (Hospital A, B and C, mentioned in the previous subsection) created on the patient-centric EHR web application were able to access and exchange the patient's data through the patient's consent by sending a data request to them and the patient was able to grant or revoke access to their medical records. The interoperability test was successful. Further, different browsers such as Chrome, Firefox, Brave, and Edge were used from the user side and users were able to exchange data without any error.

4.1.5. Discussion

For the experiment conducted we can conclude that each component of our blockchain assisted system works as expected. When a wrong private key or password was entered, the system threw an error and denied users access to their accounts. When the administrator logs in, the dashboard is accessible, and the administrator can carry out his functions as defined in the smart contract. The administrator can add hospital and hospital staff, suspend hospital accounts, and edit accounts. The hospital can send a request to a patient to access their healthcare information. When permission is given by the patient, the hospital can now view and write to the patients' record and add it to the block. This puts the patient at the center of the system and helps achieve easy data exchange by granting them custody of their healthcare records. In addition to this, data privacy is maintained. The hospital can also make appointments with the patients. Patients can view their appointment logs. The implementation of smart contracts on the blockchain gives us the level of data security required. We have shown using our security tests that users with a wrong key cannot access the data. Further, inbuilt security comes from blockchain through Keccak-256 hashes of Ethereum which makes brute-forcing address even more difficult. As this is already shown in the theoretical evaluations at [18], we have not evaluated this in our result. Further, the execution time of the smart contract is less than 300 milliseconds, which might be unnoticeable to a user (as it is less than 1 second [17]). Moreover, different hospitals created on the patient-centric EHR web application were able to access through different devices and exchange the patient's data using different browsers such as Chrome, Firefox, and Edge. The code of part of our implementation and steps to execute can be found at [19].

5. Conclusions

In this paper, our main goal is to demonstrate how blockchain technology can be used to enhance EHR systems and how our proposed model was able to solve privacy, security, and data exchange issues. For the implementation of our patient-centric EHR blockchain application, we developed a smart contract on an Ethereum consortium blockchain. Blockchain records are immutable and cannot be deleted, it ensures security, privacy and integrity. To enforce interoperability, patient data from diverse healthcare providers are maintained in a standard format and patient's consent is required for the exchange of their healthcare records. In the future, EHR may contain the electronic healthcare records of millions of people. The size of data will grow bigger and bigger with time. Therefore, in the future, a scalable system for blockchain is needed. Another problem with the proposed system is the extra overhead due to the bandwidth resources required to mine a new block and the communication that is broadcasted to all the nodes on the networks. Future work could be focused on solving those issues. Furthermore, there are several security-related works [20–22] using which the proposed framework in this work can be enhanced. In [20], a system is proposed to defend an organization, which has been already hacked by an attacker, by imposing extra costs to an attacker. In [21], a probabilistic method is used to detect the intrusions and in [22], machine learning methods are employed to detect the intrusions and to find the root cause of intrusions.

Author Contributions: Funding acquisition: A.N.; Supervision: S.S.; Writing original draft: T.F.; Writing review: A.N.; Writing review & editing: S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. de Aguiar, E.J.; Faical, B.S.; Krishnamachari, B.; Ueyama, J. A Survey of Blockchain-Based Strategies for Healthcare. *ACM Comput. Surv.* **2020**, *53*, 1–3. [CrossRef]
2. Commission Recommendation on a European Electronic Health Record Exchange Format. Available online: https://ec.europa.eu/digital-single-market/en/news/recommendation-european-electronic-health-record-exchange-format (accessed on 6 Feburary 2019).
3. Samarin, A. Exchange of Electronic Health Records Using Deposit Box Concept. 1 July 2016. Available online: http://improving-bpm-systems.blogspot.com/2016/07/electronic-health-records-ehr.html (accessed on 14 October 2020).
4. De Lusignan, S.; Mold, F.; Sheikh, A.; Majeed, A.; Wyatt, J.C.; Quinn, T.; Cavill, M.; Gronlund, T.A.; Franco, C.; Chauhan, U.; et al. Patients' online access to their electronic health records and linked online services: A systematic interpretative review. *BMJ Open* **2014**, *4*, e006021. [CrossRef] [PubMed]
5. Marcos, M.; Maldonado, J.A.; Martínez-Salvador, B.; Boscá, D.; Robles, M. Interoperability of clinical decision-support systems and electronic health records using archetypes: A case study in clinical trial eligibility. *J. Biomed. Inform.* **2013**, *46*, 676–689. [CrossRef] [PubMed]
6. Li, J. A Service-Oriented Approach to Interoperable and Secure Personal Health Record Systems. In Proceedings of the IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 6–9 April 2017; pp. 38–46.
7. Azarm, M.; Backman, C.; Kuziemsky, C.; Peyton, L. Breaking the Healthcare Interoperability Barrier by Empowering and Engaging Actors in the Healthcare System. *Procedia Comput. Sci.* **2017**, *113*, 326–333. [CrossRef]
8. Guo, Y.; Hu, Y.; Afzal, J.; Bai, G. Using P2P technology to achieve eHealth interoperability. In Proceedings of the IEEE ICSSSM11, Tianjin, China, 25–27 June 2011; pp. 1–5.
9. Gupta, S.; Sadoghi, M. Blockchain Transaction Processing. *Encycl. Big Data Technol.* **2019**, 366–376. [CrossRef]
10. Shahnaz, A.; Qamar, U.; Khalid, A. Using Blockchain for Electronic Health Records. *IEEE Access* **2019**, *7*, 147782–147795. [CrossRef]
11. Hsieh, G.; Chen, R. Design for a secure interoperable cloud-based Personal Health Record service. In Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, Taipei, Taiwan, 3–6 December 2012; pp. 472–479.
12. Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. MedRec: Using blockchain for medical data access and permission management. In Proceedings of the IEEE International Conference on Open and Big Data (OBD), Vienna, Austria, 22–24 August 2016; pp. 25–30.
13. Xia, Q.; Sifah, E.B.; Smahi, A.; Amofa, S.; Zhang, X. BBDS: Blockchain-based data sharing for electronic medical records in cloud environments. *Information* **2017**, *8*, 44. [CrossRef]
14. Zhou, L.; Wang, L.; Sun, Y. MIStore: Blockchain-Based Medical Insurance Storage System. *J. Med. Syst.* **2018**, *42*, 1–17. [CrossRef] [PubMed]

15. Alevtina, D.; Xu, Z.; Ryu, S.; Schumacher, M.; Wang, F. Secure and Trustable Electronic Medical Records Sharing using Blockchain. In Proceedings of the Annual Symposium Proceedings, AMIA Symposium, Washington, DC, USA, 4 November 2017; pp. 650–659.
16. Eberhardt, J.; Tai, S. On or Off the Blockchain? Insights on Off-Chaining Computation and Data. In *Service-Oriented and Cloud Computing. ESOCC 2017*; De Paoli, F., Schulte, S., Broch Johnsen, E., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10465. [CrossRef]
17. Nielsen, J. *Usability Engineering*; Morgan Kaufmann: San Francisco, CA, USA, 1993; ISBN 0-12-518406-9.
18. Kościelny, C.; Kurkowski, M.; Srebrny, M. An Electronic Signature and Hash Functions. In *Modern Cryptography Primer: Theoretical Foundations and Practical Applications*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 127–145.
19. Our Implementation Code. Available online: https://github.com/Tomilayohub/hospital_records (accessed on 5 Feburary 2021).
20. Chakraborty, T.; Jajodia, S.; Katz, J.; Picariello, A.; Sperli, G.; Subrahmanian, V.S. FORGE: A Fake Online Repository Generation Engine for Cyber Deception. *IEEE Trans. Dependable Secur. Comput.* **2019**. [CrossRef]
21. Albanese, M.; Erbacher, R.; Jajodia, S.; Molinaro, C.; Persia, F.; Picariello, A.; Sperlì, G.; Subrahmanian, V.S. Recognizing Unexplained Behavior in Network Traffic. In *Network Science and Cybersecurity*; Springer: New York, NY, USA, 2014; pp. 39–62.
22. Abdelkefi, A.; Jiang, Y.; Sharma, S. SENATUS: An Approach to Joint Traffic Anomaly Detection and Root Cause Analysis. In Proceedings of the 2018 2nd Cyber Security in Networking Conference (CSNet), Paris, France, 24–26 October 2018.