A Secure Decentralized Privacy-Preserving Healthcare System Using Blockchain

Aderonke Thompson, Hafiz Odekunle, and Boniface Alese

1 Introduction

Healthcare is a concentrated knowledge space in which vast quantities of information are processed, accessed, and distributed all the time [2]. Storing and sharing a massive number of records is necessary and challenging due to the sensitivity of health information and others that restrict data, regarding security and privacy. In the healthcare field, knowledge exchange is vitally essential for diagnosis and decision making. Information exchange is vital for medical professionals to be able to communicate with each other and pass patient information to the competent authority for other purposes, such as testing. The exchange of medical information must be achieved across a safe network and must also ensure that the privacy of patients is maintained.

Healthcare blockchain is an innovation that helps to safely customize stable health data, share it by blending the complete real-time information of a patient's health, and store it as a secured healthcare arrangement [5]. This technology permits transactions by participants in dispersed, permanent, straightforward, secure, and auditory ways, which allow access to records from the first transaction, which can be confirmed and examined by any entity. The chain is continually developing and new blocks holding references, that is, a hash value is being added to the existing block [3]. Blockchain is structured as a peer-to-peer (P2P) network that links with numerous network nodes. All the nodes in the system have a public key and a private key for securing information exchange. Blockchain transaction uses a public key for encryption to guarantee consistency, irreversible, and non-reputability of records while decryption of the message uses the private key for integrity and verification of

A. Thompson (✉) · H. Odekunle · B. Alese
Federal University of Technology, Akure, Nigeria
e-mail: afthompson@futa.edu.ng; bkalese@futa.edu.ng

K. Daimi et al. (eds.), *Principles and Practice of Blockchains*,
https://doi.org/10.1007/978-3-031-10507-4_9

every transaction made by node [5]. In this regard, only the public key and private key authentic messages go to the network for affirmation [1]. The challenge of the technique is that solitary clients with a particular private key are permitted to sign the transaction. Also, mistakes during transmission of the information lead to system failure, for example, confirming an advanced signature. The transactions that are considered legitimate are broadcasted in the network domain by the miners. The miners decide data transactions to admit in the distributed public ledger based on the chosen consensus protocol used, for examples proof-of-work (PoW) and proof-of-stake (PoS). The approval nodes check that the communicated block encompasses large transactions and references the former block in the chain utilizing the matching hash value. Thus, attaining these requirements implies that the new blocks are added to the blockchain; otherwise, it drops the block.

2 Overview and Related Work

There are different types of blockchains based on the managed data, on the accessibility of such data, and on what operation can be performed by the user. These include public permissionless, consortium (public permission), and private.

Public permissionless: This is a state-of-the-art public blockchain protocols based on proof-of-work (PoW) consensus algorithms with open source and not permission. Anyone can participate as a node or miner without permission. All data in the blockchain is accessible and visible to everyone, although parts of the blockchain can be encrypted to secure data and preserve user's anonymity. Examples are Bitcoin, Ethereum, or Litecoin.

Consortium (public permissioned): This type of blockchain operates under the leadership of a group. As opposed to public blockchain, they do not allow any person with access to the Internet to participate in the process of verifying transactions; only a selected group of nodes can participate in the distributed consensus process. It is used within one or across many institutions. When a consortium blockchain is created within one institution (e.g., financial sector), it is initiated for restricted public use and fractionally centralized. On the other hand, a consortium between institutions (e.g., insurance companies, financial institutions, governmental institutions) is unlocked for public use while still having created a relatively centralized trust.

Private: In a private blockchain, write permissions are kept centralized to one trusted organization, while read permissions may be public or restricted. A private blockchain only allows selected nodes to connect to the network. It is, therefore, yet a distributed centralized network. Private blockchains control which nodes can perform transactions, execute smart contracts, or act as miners. It is used for private purposes. Hyperledger Fabric and Ripple are examples of blockchain platforms that only support private blockchain networks. Table 1 presents the summary of the types of blockchain.

Table 1 Difference between public, consortium, and private blockchain

	Public	Consortium	Private
Participants	Permissioned Identified Trusted	Permissioned Identified Trusted	Permissioned Identified Trusted
Consensus Mechanisms	Proof of Work, Proof of Stake, etc. Large energy consumption. No finality 51% attack	Voting or multi-party consensus algorithm Lighter Faster Low energy consumption Enable finality	Voting or multi-party consensus algorithm Lighter Faster Low energy consumption Enable finality
Access	Open Read/Write	Permissioned Read and /or Write	Permissioned Read and /or Write
Transaction Approval Frequency	Long Bitcoin: 10min or more	Short	Short
Speed	Slower	Faster	Faster
Security	Proof of Work Proof of Stake Other Consensus Mechanisms	Approved participants	Approved participants

2.1 Consensus Protocols

In a world where trust is expensive, it is essential to understand the unstable nature of trust and to figure out some measures of consensus among ourselves in respect to that which we hold as "truth." For the blockchain network to continue as functional, its peers need to come to terms with a specific state of the distributed ledger and on a way for storing data into blocks. Such terms are known as a distributed consensus protocol and it affirms the chronological order of generated transactions.

2.1.1 Proof-of-Work (PoW)

PoW is carried out by miners who conducted through miners competing to solve a cryptographic problem—also known as a hash puzzle. These miners help to verify every Bitcoin transaction, where it involves producing a hash-based PoW that is based on previous transaction blocks (read up on the Merkle Tree for more information) and forms a new branch with a new transaction block. This means that the work is moderately difficult for the miners to perform but easy for the network to verify. The first miner who manages to produce the PoW is awarded some Bitcoins. Over time, the amount of Bitcoin awarded decreases.

2.1.2 Proof-of-Stake (PoS)

Unlike PoW where new transaction blocks are created based on computational work done by solving a complex cryptographic puzzle, PoS allows a forger (instead of a miner) to stake any amount of cryptocurrency held, to be probabilistically assigned a chance to be the one validating the block—the probability based on the amount of cryptocurrency staked. Additionally, for most PoS systems, instead of receiving a cryptocurrency reward (in the above case, the Bitcoin miners receives some Bitcoins for solving a PoW), the forgers instead take the transaction fees as rewards.

The idea of putting coins to be "staked" prevents bad actors from making fraudulent validations—upon false validation of transactions, the amount staked will be forfeited. Hence, this incentivizes forgers to validate legitimately. In the recent year, PoS has gained attention, with Ethereum switching towards a PoS from a PoW consensus system.

2.1.3 Delegated Proof-of-Stake (DPoS)

DPoS is similar to PoS in regard to staking but has a different and a more democratic system that is said to be fair. Like PoS, token holders stake their tokens in this consensus protocol. Instead of the probabilistic algorithm in PoS, token holders within a DPoS network are able to cast votes proportional to their stake to appoint delegates to serve on a panel of witnesses—these witnesses secure the blockchain network. In DPoS, delegates do not need to have a large stake, but they must compete to gain the most votes from users.

It provides better scalability compared to PoW and PoS as there are fully dedicated nodes who are voted to power the blockchain. Block producers can be voted in or out at any time, and hence the threat of tarnishing their reputation and loss of income plays a major role against bad actors [10]. No doubt, DPoS seems to result in a semi-centralized network, but it is traded off for scalability.

Like PoS, DPoS has also gained attention over the years with several projects adopting this consensus algorithm. Since it was invented by Dan Larimer, DPoS has been refined continuously, from BitShares to Steem and now in EOS.

2.1.4 Proof-of-Authority (PoA)

PoA is known to bear many similarities to PoS and DPoS, where only a group of preselected authorities (called validators) secure the blockchain and can produce new blocks. New blocks on the blockchain are created only when a super majority is reached by the validators. The identities of all validators are public and verifiable by any third party—resulting in the validator's public identity performing the role of proof-of-stake. As these validators' identities are at stake, the threat of their identity being ruined incentivizes them to act in the best interest of the network.

Since PoA's trust system is predetermined, concerns have been raised that there might be a centralized element with this consensus algorithm. However, it can be argued that semi-centralization could actually be appropriate within private/consortium blockchains—in exchange for improved scalability. Newer blockchain start-ups have ventured into implementing PoA. In addition, Ethereum testnets like Rinkeby and Kovan explores the use of a PoA consensus algorithm.

2.1.5 Access Control Mechanism with Smart Contract for Data Sharing

Sharing healthcare data is considered to be a critical approach to improve the quality of healthcare service and reduce medical costs. Though current EHR systems bring much convenience, many obstacles still exist in the healthcare information systems in practice, hinder secure and scalable data sharing across multiple organizations, and thus limit the development of medical decision making and research [7]. From the foregoing, there are risks of the single-point attack and data leakage in a centralized system. Besides, patients cannot preserve the ownership of their own private data to share with someone who they trust. It may result in unauthorized use of private data by curious organizations. Furthermore, different competing organizations lacking partnership trust are not willing to share data, which would also hinder the development of data sharing [9].

In this case, it is necessary to ensure security and privacy protection and return the control right of data back to users in order to encourage data sharing. It is relatively simply to deal with security and privacy issues when data reside in a single organization, but it will be challenging in the case of secure health information exchange across different domains. Meanwhile, it also needs to further consider a suitable technique to boost efficient collaboration in the medical industry.

Securing access control mechanism as one of the common approaches requires that only authorized entities can access sharing data. This mechanism includes access policy commonly consisting of access control list (ACL) associated with data owner. ACL is a list of requestors who can access data and related permissions (read, write, update) to specific data. Authorization is a function of granting permission to authenticated users in order to access the protected resources following predefined access policies. The authentication process always comes before the authorization process. Access policies of this mechanism mainly focus on who is performing which action on what data object for which purposes. Traditional access control approaches for EHRs sharing are deployed, managed, and run by third parties. Users always assume that third parties (e.g., cloud servers) perform authentication and access requests on data usage honestly. However, in fact, the server is honest but curious. It is promising that combining blockchain with access control mechanism is to build a trustworthy system. Users can realize secure self-management of their own data and keep shared data private. In this new model, patients can predefine access permissions (authorize, refuse, revoke), operation (read, write, update, delete), and duration to share their data by smart contracts on the blockchain without the loss of control right. Smart contracts can be triggered on the blockchain once all of preconditions are met and can provide audit mechanism for any request recorded in the ledger as well. There are many existing studies and applications applying smart contract for secure healthcare data sharing. A study proposed that patients can authorize access to their record only under predefined conditions (research of a certain type, and for a given time range) [15].

Smart contract placed directly on the blockchain verifies whether data requestors meet these conditions to access the specified data. If the requestor does not have the access rights, the system will abort the session. Similarly, smart contracts in a study [4] can be used for granting and revocation of access right and notifying the updated information as providers move in and out of networks.

Researchers designed a decentralized record management system based on blockchain, called MedRec [1]. In this system, patient-provider relationship contract is deployed between any two nodes in which patients manage and share medical records with healthcare providers. Providers can add or modify this record in the case of patient's permissions. Data access record is preserved in the block to track the malicious entities when violated access activities happen. They also designed a simple graphical interface tool that allows patients to share off-chain data with fine-grained access control.

The similar design is proposed in [13]. Researchers developed an access protocol based on smart contract through admin component when mobile users send the request [12]. Smart contract will verify any transaction by predefined policies of access protocol to prevent malicious attack and achieve reliable EHRs sharing. But curious miners may infer personal information during the mining process due to the processing transactions including area ID, mobile gateway ID, and patient ID. A study creatively adopted the channel scheme of Hyperledger Fabric, which separates different types of activities for users in the different channels to share different grained data [8]. Chaincode (smart contract) can be launched in the channel with different access type, permissioned operations, and selective shared data specified in the certificate by data owners. In addition to data sharing, such a channel scheme makes good use of Fabric to enhance data privacy.

2.2 Smart Contract

A smart contract is a tamper-proof computer program or protocol that can verify and execute itself. Nick Szabo comes up with the idea of smart contract in 1994. It allows executing code without the third parties. A smart contract comprises of the value, address, functions, and state. It accepts transaction as an input and triggers event as the output after executing the corresponding code. Implementation of function logic determines the state of the contract. The necessity of smart contract has been the major focus since the emergence of blockchain technology in 2008 when the technology comes into existence through Bitcoin cryptocurrency because it has the capacity to publicly maintain database and peer-to-peer transactions securely and create a trustful environment. Smart contracts are auditable and irreversible. All the transaction information is present in a smart contract, and it executes automatically. A smart contract is machine readable, an event-driven program, autonomous, and distributed.

Solidity is a high-level language used to implement smart contracts. Developing blockchain platform of solidity are Ethereum, ErisDB, Zeppelin, and Counterparty. According to Nick Szabo, the contractual clauses (collateral, bonding, delineation of property rights) should be encoded and embedded in the required hardware and software. This helps to minimize the requirement of any trusted third party for communication using smart contracts and at the same time makes the system secure against any malicious attack. In the case of blockchain-based smart contracts, contracts are nothing but scripts residing on the blockchain, which has the ability to execute them. One can trigger a transaction to a smart contract by using the unique addresses assigned to it by the blockchain technology. Let us take an example to better understand the working of smart contracts. Suppose you want to sale your house or rent your apartment to someone, then you can simply deploy a smart contract in an existing blockchain network. Information regarding the property can be stored in the blockchain and anyone belonging to that network can access that information, but they cannot change it. In this way, you can find a buyer for your property without the need of any third party. For a wide range of potential applications, blockchain-based smart contracts could offer a number of benefits:

- Speed and real-time updates
- Accuracy
- Lower execution risk
- Fewer intermediaries
- Lower cost
- New business or operational models (Fig. 1)

With the technology growing fast, human living standard also grows vastly. In the recent use of developed devices and supporting technology, human can monitor his or her health condition just sitting at home. There are lots of devices that are already developed to read different attributes in the human body. These data can be collected using low-end device and processing locally to get quick information [17]. Blockchain technology helps to maintain the privacy of the patients and maintained

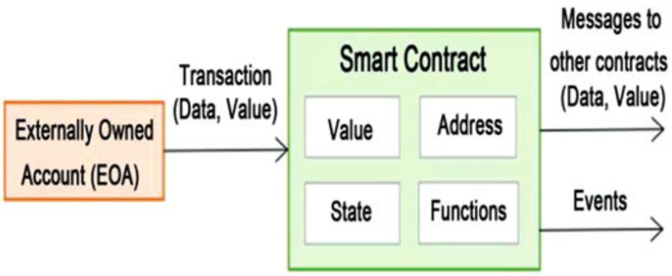

Externally Owned Account (EOA)
Transaction (Data, Value)
Smart Contract
Value
Address
State
Functions
Messages to other contracts (Data, Value)
Events

Fig. 1 Structure of smart contract

data in digital ledger format. A smart contract can be used in that system to make the system more reliable and automated. Using a smart contract, human can write some terms and conditions which could be applied once data are collected. Then it will execute these smart contracts and trigger corresponding events.

The blockchain has the capability to boost data sharing in healthcare. Researchers provide an easier way for patients to govern their medical data by developing App HGD (Healthcare Data Gateway) which is built in blockchain technology [14]. All data are managed using data management layer and stored on blockchain cloud. Patients' data are accessible solely via an authorized user; in addition, data replica may be enforced to be destroyed when the authorized time elapsed.

A study also proposed an exchange network approach for health information [15]. The two contributions are as follows: it makes use of electronic health record (EHR) semantic and design checking to organize all EHRs in the blockchain network to solve the issue of interoperability. Also, an algorithm is proposed to select the next miner randomly to reduce the power and system resources used in computation of POW. In addition, it is suggested that privacy and anonymity can be provided using blockchain encryption, smart contracts, and privacy-preserving keyword searches, but detailed approaches were not included.

Researchers proposed a medical system called MedRec which was based on smart contracts for effective and simpler administration of EHRs [2]. Registrar contracts (RCs) are used to map the user's identification string to their Ethereum addresses to keep anonymity of the users. Summary contracts (SCs) contained links that referenced patient-provider relationships (PPRs) to ensure that all the medical records of patients are connected. How patients' data are managed and accessed is defined in PPRs. However, the detailed approaches to solve an issue such as how to encrypt the patients' EHR, accessed by authorized users, user authentication, etc.

A study proposed a blockchain and MedRec-based way to deal with unravel security issues such as confidentiality, access control, privacy, audibility, and integrity in sharing healthcare data [16]. The barriers are tackled by using a signcryption and attribute-bases authentication (ABA) to ensure that process of data sharing is secure. The proposed model provides the following services: (1) Data authenticity—the validness of patients' EHRs can be confirmed by who access the information. (2) Data integrity—it ensures that stored patients' EHRs are guided against altering. (3) Data confidentiality—patients' EHRs are stored securely and stayed discreet from the unapproved user.

Researchers adopt Ethereum blockchain smart contracts to achieve efficient collaboration, data integrity, and protection of patient privacy. Protecting healthcare professional's privacy and securing links to establish an interoperability end-to-end reachable network among independent healthcare system are provided by [9]. The research solves the problem by storing the patient health records in a secure off-chain database and makes a secure socket to trade authorization-based access to tolerant information utilizing standard public key cryptography.

A study formulated a data sharing mechanism using Blockchain [9]. The mechanism was made up of three components, namely, client, control system, and storage. It is an efficient, secure identity-based authentication and key agreement protocol.

An effective security measure based on personality validation and key understanding protocol is adopted to assure user anonymity and authenticity. Keys were generated to execute client confirmation and enrollment, client verification, and request creation. A study proposes a structure that characterizes some authorization access rules through the Hyperledger Fabric [11], where a service provider can access medical record of patients in a crisis condition under the limitations of patient's permission through the system. Smart contract handled authorization and fetching of data of all transactions from ledger which makes the framework secured, efficient, and auditable.

3 Methodology

Blockchain is invented for storing financial-related records and provides a structure for actualizing a decentralized system. Each node of the blockchain interacts with another via cryptographically encrypted information exchange, and the transaction-based state system of the blockchain is known as smart contracts [6]. It was first fully implemented by Ethereum. A smart contract is client rights management tools that provide coordination and enforcement frameworks for network participant agreements without traditional legal contracts being required [7]. For instance, a smart contract characterizes the application logic that executes at whatever point an exchange happens in the trading of digital money.

Ethereum smart contracts create intelligent and logical representations of existing health data stored on each node on the network. The smart contract contains the metadata of a patient's data, access permission, and data integrity. The contract enforces that each transaction made on the blockchain system conveys a cryptographically signed information to oversee these traits. Contracts implement approaches to create data exchanges by valid blockchain transactions only and execute any set of rules consigning a specific health record; so far it can be processed.

3.1 System Design

The design integrates each node to an existing EHR system. Each node, specifically representing healthcare providers, is expected to have already databases with medical records saved on servers connected to the network.

The system design is made up of three layers, namely, data collection layer, data repository layer, and data sharing layer.

Data collection layer In this layer, electronic health records (HER) are generated by a service provider, i.e., a medical doctor. The EHRs are signed by a medical doctor using content extraction signature (CES) scheme and the signed EHRs are forwarded to the patients. Patients can extract sensitive information of EHR and create authentic extraction signature in order to avert privacy in case there is leakage during the data sharing.

Data repository layer This layer is responsible for storing the EHR and its indexes. The layer comprises of the following components:

1. *EHR manager*: This represents the different health institutions in the local system; these local systems are connected to blockchain network; the primary role of the system in our design is to store the EHRs and generate indexes that points to the stored EHR. The EHR manager consist of the following:
   (a) API library: API library will manage and control the operation of the system. The API executes a function call to interact with blockchain by parsing blockchain protocol to connect with an Ethereum client.
   (b) Ethereum client: This operates an extensive set of tasks, which include peer-to-peer network connection, encoding transactions, and transmitting transactions in addition to keeping a verified local copy of the blockchain.
   (c) Database manager: This manages access to the node's local database and ensures permission governance storage on the blockchain. The database manager checks the blockchain contracts for address verification request which is acceptable to query access. On verifying the address, the query is run on the node's local database and then results are returned to the client.
2. *Blockchain server*: This serves like a cloud server, which is responsible for saving and communicating the electronic health records (EHRs). The EHR storage location links and predefined access permission (smart contracts) of the patients are stored in the blockchain. The smart contract ensures that data are shared securely and records each access request and activities in the blockchain for the purpose of auditing in the future (Fig. 2).

Data sharing layer The authorized users (patients, healthcare providers, researchers) can initiate a request for patients' EHRs which can be granted or denied based on the predefined conditions set by the patients. Also, patient can revoke access given to a user at any given time. The logical and intelligent access control is designed using Ethereum's smart contract. The contract contains the storage link of patient's EHRs, access permission, and timestamp. The contract enforces that each transaction made on the blockchain system carries a cryptographically signed instructions to manage data integrity. The smart contract is structured as follows:

(a) Identity catalog contract (ICC): Identity catalog contract is one of the major contracts deployed on electronic health record management system (EHRMS); this contract only occurs once and invokes by every node present on EHRM network. It maps patient identity strings to their Ethereum address public key and provider IP address (Fig. 3).

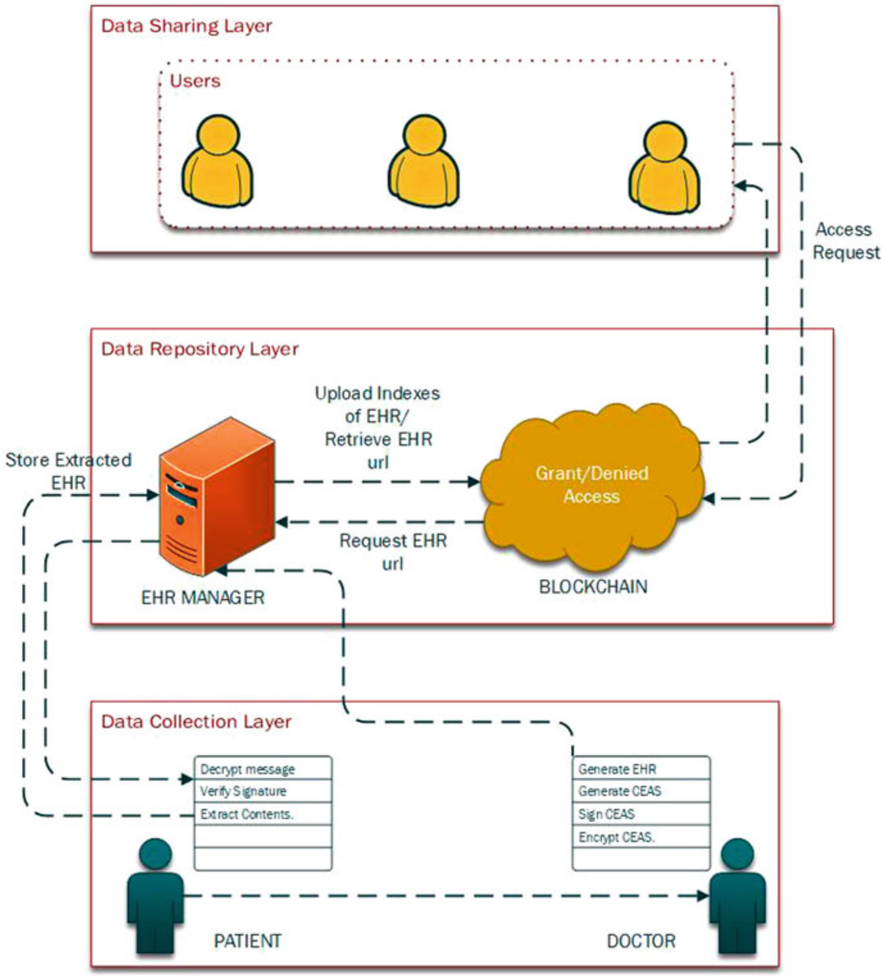


Fig. 2 Proposed system design

(b) Relationship contract (RC): The relationship contract established a patient-provider relationship (PPR) between all entities who have access to EHR of a particular patient. For each contract, one patient and provider are connected to it. The PPR is also made up of a list of unlimited third parties who are granted permission to only view the patient EHR. Patient, provider, and third-party relationships are stored on the blockchain and enable the third party to locate all providers in EHRMS network that has access to a particular patient's EHR instead of searching for them one after the other. EHRMS enables the patient to have total control over who accesses their EHR and complete audits of their EHR which are being accessed through PPR contracts.

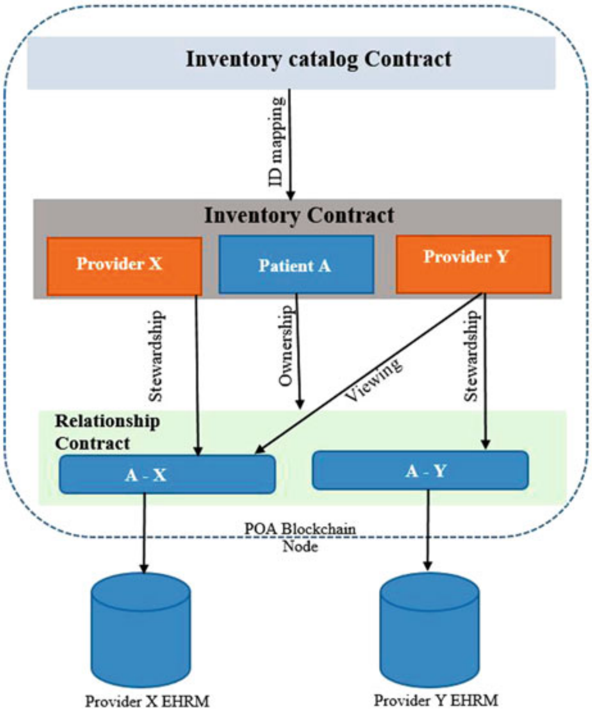


Fig. 3 Relationship graph between contract and network nodes

(c) Inventory contract (IC): IC holds a list of references to RC, representing all the participant's previous and current transactions with other nodes in the system. The contract represents actors in EHRM. Patients communicate the system by utilizing an inventory contract as a proxy. Information about entities present in the system are stored in these contracts. Information stored on the contract include patient-provider relationship (PPR) and some addresses that are associated with the individual account. These contracts require only for the patient as it serves as a pointer to all PPR that is stored in one place on the blockchain and also for easier retrieval PPR if the patient needs to recover their account using recovery secrete text.

3.2 Content Extraction Signature

In this work, document processing operation call extraction is used to remove certain selected part of the signed EHR before it is stored in the EHR manager and available for user on the blockchain network. This is done to revoke sensitive data that prevent anonymous participation of patients on the blockchain. Content extraction signature allows a patient P, the owner of EHRs which was signed by a doctor D, to extract part of EHRs and send only those parts to a user U, which helps us to assure the patient's privacy.

The content extraction signature (CES) scheme can be defined as follows: CES = (GK, Sig, Ext, Ver) which represents four algorithms, namely, key generation algorithm, signature generation algorithm, signature extraction algorithm, and signature verification algorithm.

3.2.1 Key Generation GK(k)

This algorithm chooses a security parameters k and generates a public/private key pair (pk, sk).

Pick a large prime number P and a generator g in Z_p

```
$y \leftarrow pg$;
$Q \leftarrow (p-1)(g-1)$;
Select a random number q in such a way that gcd($q, Q$) = 1.
Compute:
$d \leftarrow q^{-1} mod y$;
Publish Public Key $pk \leftarrow (y,q)$ and Private Key $pk \leftarrow (y,q,d)$
Algorithm 2
```

3.2.2 Signature Generation Algorithm Sig (SK, M, CEAS)

This algorithm accepts private key (sk), EHRs document (M), and a content extraction signature structure CEAS and produce content extraction signature σ_F.

```
Parse Doctor private key sk ← (y, q, d); Patients EHR, M and
Content Extraction Signature Structure CEAS
$N \leftarrow Len(M)$;
$T \leftarrow \{0,1\}$;
For i $\epsilon [n]$
$h[i] \leftarrow H(CEAS||T||i||M_i)$;
$\sigma[i] \leftarrow h[i]^d mod\ y$;
$\sigma_F \leftarrow H(CEAS||T||<\sigma[i]>_{i\epsilon[n]})$;
Return $\sigma_F$
```

3.2.3 Signature Extraction Algorithm Ext(pk, M, σ_F, X)

This algorithm accepts a public key *pk*, a EHR document *M*, a Content extraction signature σ_F, an extraction subset *X*, and produce an extracted signature σ_E.

```
Parse $pk = (y,q)$
Parse $\sigma_F \leftarrow H(CEAS||T||<\sigma[i]>_{i\epsilon[n]})$
$\sigma \leftarrow \prod_{i \in X} \sigma[i]\ mod\ y$
$\sigma_E \leftarrow (CEAS, T, \sigma)$
Return $\sigma_E$
```

3.2.4 Signature Verification Algorithm $Ver(pk, M', \sigma_E)$

This algorithm accepts a public key pk, an extracted subdocument M', and an extracted signature ,σ_E, and produces a verification decision $d \in \{Acc, Rej\}$, where Acc implies "Accept" and Rej indicates "Reject."

```
Parse $pk = (y,q)$
Parse $\sigma_E \leftarrow (CEAS, T, \sigma)$
Parse $M^1$
$X' \leftarrow CI(M'): n \leftarrow len(M')$
For $i \in X'$
$h[i] \leftarrow H(CEAS, T, n, i, M'[i])$
if
$\sigma^q \equiv \prod_{i \in X'} h[i] (mod\ y)$ and $X' \in CEAS$
Return Acc
Else Return Rej
```

4 Implementation

The designed system, electronic health record management system (EHRMS), works by joining three components together: (1) a front end which is a web application that allows patients/provider to access EHRMS; (2) an EHR manager that communicates with provider databases, and file system; and (3) an Ethereum blockchain that controls access rights to EHRM and connects each EHR manager to the peer-to-peer network. In summary, patients initiate access contracts that are saved on the blockchain. Therefore, the contracts determine providers' actor that should be permitted to access the EHR.

4.1 Front End

This is a component of the system that the user interacts with. All patients and service providers communicate with the system through this interface. The front end was developed using React framework. React is a JavaScript library for building a user interface with simple views for each state of the application; it communicates to the local and remote server through WebSockets. Users' private keys and other details are stored on local machines while providers use a remote server to store information of users that have a relationship with them (Figs. 4 and 5).

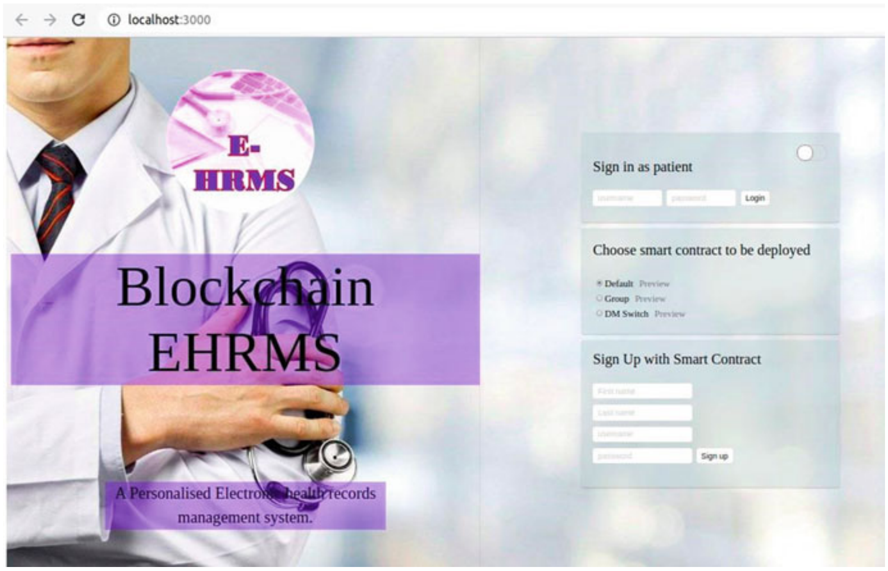
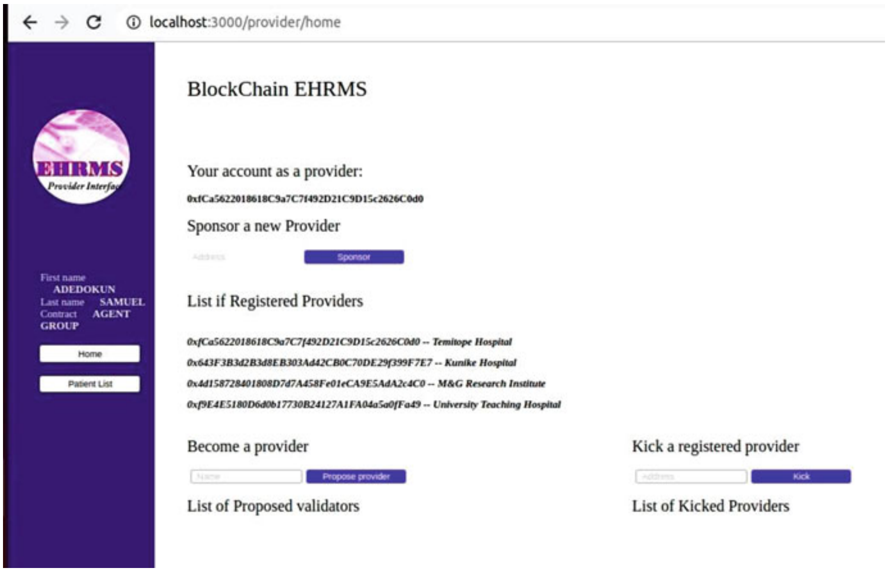


Fig. 4 EHRMS front end landing page



BlockChain EHRMS

First name ADEDOKUN
Last name SAMUEL
Contract AGENT GROUP

Your account as a provider:

0xfCa5622018618C9a7C7f492D21C9D15c2626C0d0

Sponsor a new Provider

List if Registered Providers

0xfCa5622018618C9a7C7f492D21C9D15c2626C0d0 -- Temitope Hospital
0x643F3B3d2B3d8EB303Ad42CB0C70DE29f399F7E7 -- Kunike Hospital
0x4d158728401808D7d7A458Fe01eCA9E5AdA2c4C0 -- M&G Research Institute
0xf9E4E5180D6d0b17730B24127A1FA04a5a0fFa49 -- University Teaching Hospital

Become a provider

List of Proposed validators

Kick a registered provider

List of Kicked Providers

Fig. 5 Service provider home page

4.2 EHR Manager

The EHR manager segment of EHRMS connects with the basic file system and facilitates correspondence between EHRMS hubs. It is essentially written in GoLang, albeit a few collaborations with the blockchain network require script written in JavaScript.

Remote procedure calls (RPCs) are orders given by one PC to conjure functions on another. The protocol is transport rationalist and can be applied to the Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP), WebSocket, and different protocols for sharing information over the web. EHRMS utilizes RPC in two different ways, between the users and a local database, and between the users and a remote provider database.

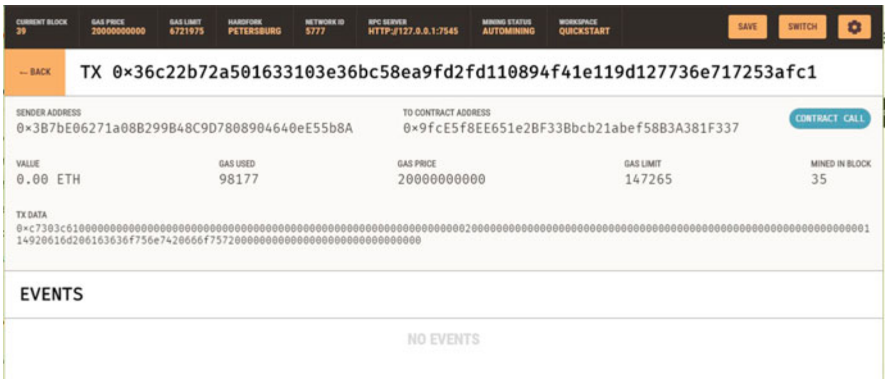
Local RPCs are utilized to interface between parts of the user interface written in JavaScript and the EHR manager written in GoLang. The EHR manager is fundamental for composing client information to the host file system. This incorporates usernames, passwords, and secret texts.

4.3 Ethereum Proof-of-Authority

Blockchain network was set up with three different systems. Each system represents the provider node, and each node was set into motion as a validator to the genesis block on the proof-of-authority (PoA) blockchain.

PoA is an algorithm the makes use of consensus mechanism that relied on the identity at stake. In PoA, an authorized account known as validator approved all the transactions and blocks chained into the network (Fig. 6).

In EHRMS, the validator provides a service known as the Ether Faucet, and the Ether Faucet is the fulfillment of the value of the ether required to perform


CURRENT BLOCK 39 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK PETERSBURG | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

TX 0×36c22b72a501633103e36bc58ea9fd2fd110894f41e119d127736e717253afc1

SENDER ADDRESS
0×3B7bE06271a08B299B48C9D7808904640eE55b8A

TO CONTRACT ADDRESS
0×9fcE5f8EE651e2BF33Bbcb21abef58B3A381F337

VALUE
0.00 ETH

GAS USED
98177

GAS PRICE
20000000000

GAS LIMIT
147265

MINED IN BLOCK
35

TX DATA

```
0×c7303c6100000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000001
14920616d206163636f756e7420666f7572000000000000000000000000000000
```

EVENTS

NO EVENTS

Fig. 6 Smart contract calls on the Ganache platform

a transaction on the network. Because of convenience, this process is obscure for system users. EHRMS requests for enough ether necessary to carry out transactions from a provider, for all transactions made by a user. This usually happens when patients perform their contract with an agent or when a provider becomes a validator. Providers that are validators automatically generate ether, as nodes endorse the blocks. Three specific types of smart contracts have been created, compiled, and deployed on the PoA blockchain to handle access control to EHR using a solidity programming language. The smart contracts are summary contracts, contracts for patient-provider relationships (PPRs), and contracts for registrars contracts. The compiler generates byte code, and each byte code is uploaded to blockchain EVM to represent a specific operation. The compiler generates byte code, and each byte code is uploaded to blockchain EVM to represent specific operation.

4.4 Storing EHR

A patient (P) has an encounter with a medical doctor (D) and the D generates a health record as EHRs for P. The D sends EHRs to P; upon acquiring the EHRs, P uploads the indexes of the EHR to the blockchain with the catalog of authorized user (U).

EHRMS allows each user to create a pair of keys during registration, the doctor (D) creates content extraction on EHRs using a pair of keys (pk, sk) and encrypt EHRs using symmetric key $\mathrm{k_d}$ for the purpose of confidentiality. All users in the blockchain network possess public key ($\mathrm{pk_i}$) and private key ($\mathrm{sk_i}$) to accomplish data sharing.

In our system, CES scheme is used to extract sensitive information from EHR to assure patient privacy. The doctor serrates EHRs into nine parts (name, age, sex, ID, contact, next of kin, medical history, diagnosis, prescription), and it is denoted as $M = \{m_1, m_2, m_3,\ m_4,\ m_5, m_6,\ m_7,\ m_8,\ m_9\}$. Then the doctor defines the content extraction access structure $CEAS = \{2, 3, 7, 9\}$ to avert virulent extraction. The process of content extraction signature is started by generating public/private key (pk, sk) using key generation algorithm. Next is for the D to sign the EHR data M using the signature extraction algorithm. After the completion of signature algorithm, the D encrypts EHR $(M, hi_{i\,\in\,[1,9]}, \sigma_F, CEAS, T)$ and patient's public key pk_p and sends both encrypted data to patient P.

$$\text{Message} = \left\{E_{k_d}\left(M, hi_{i \in [1,9]}, \sigma_F, CEAS, T\right), E_{pk_p}(k_d)\right\} \tag{1}$$

Upon receiving the encrypted data from D, P decrypt k_d and get M. The P uses CEAS to extract content of the signature and the corresponding EHR using signature extraction algorithm.

$$\text{Data} = \{(M_i, h_i, T), \sigma_E\} \tag{2}$$

P stores the extract signature and corresponding EHR in EHR manager and returns the storage location link. P partakes in the blockchain and anonymously by presenting the signed storage location link $\mathrm{SIG}_{\mathrm{skp}}(\mathrm{Ind})$, and then a request (Rq) is sent to the blockchain where $i \in CI(M^{'})$ and t is timestamp.

$$Rq = \left\{SIG_{sk_p}\left(Ind_i\right), H\left(Ind_i\right), E_{pk_p}\left(Ind_i\right), t\right\} \tag{3}$$

$$Ind_i = (link_i, h_i, t) \tag{4}$$

After receiving, the request validator checks the validity of each transaction and collates all data during the period into a data set $Data_{set} = \{Rq, t\}$. The validator forms a new data block by hash the data set $Data_{hash} = H(Data_{set}, t)$, and digital signature $SIG_{sk_v}\Big(Data_{set}, Data_{hash}$. The validator broadcasts the data block to all the nodes on the blockchain.

$$Validator \rightarrow All : D_{block} = \left\{Data_{set,} Data_{hash,} pk_{v_i}, SIG_{sk_{vi}}\left(Data_{set}, Data_{hash}\right)\right\} \tag{5}$$

4.5 Sharing of EHR

To ensure save sharing of EHRs, P predefines access permission in the smart contract, i.e., access right and access activity, for instance, write, read, and duration of access. Smart contract is automatically prompted to run the corresponding operation once the predefined condition set is met. Data sharing process starts by initiating an EHR sharing request transaction (Rq) to the blockchain network. User U makes this request which includes the access target (ID), object to access (obj), and access content. Validators accept the transaction request and verify the U identity. If U is valid, the transaction will be stored in the blockchain.

$$U \rightarrow Validator : Rq = (ID, obj, ind, t), ind \in [1, 9] \tag{6}$$

If access conditions are met by the request, smart contract is prompted to decrypt the indexes of EHRs with patient private key skp and retires the cipher-text of indexes to U: else the request is rejected.

$$Message = E_{pk_u}\left(\mathrm{i}nd_i, t\right) \tag{7}$$

Table 2 Key recovery query execution time with varying database size

Key recovery/DB size	100 (ms)	1000 (ms)	10,000 (ms)
1	1.75	1.78	1.84
10	1.80	2.20	2.13
100	2.00	7.80	9.88
1000	4.38	11.00	13.12

5 Evaluation

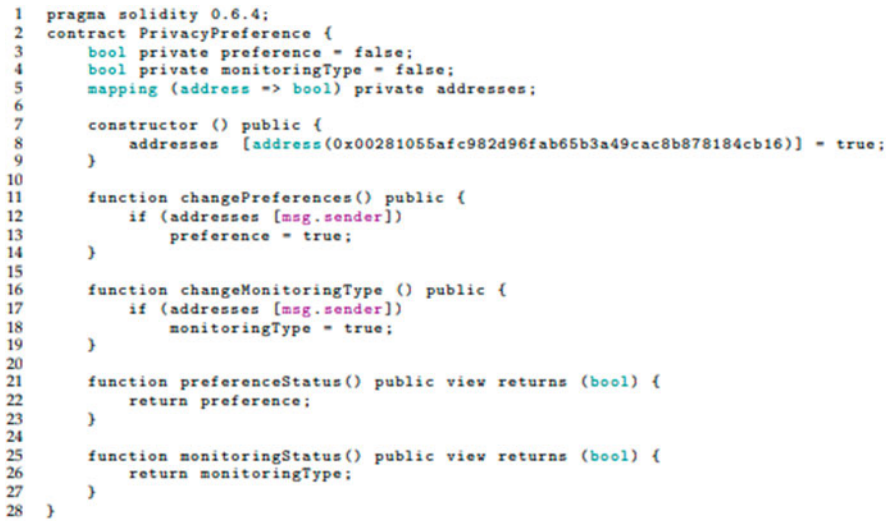
This section gives a performance study based on a series of experiments conducted to evaluate our system (EHRMS). We will first explain how the public key is obtained.

The first test was to determine a user's public key retrieval time connecting the EHR manager (gateway) database. We employed MySQL database system in the tests, on a host PC with Debian 9.8 OS, 4 GB RAM, and an Intel Core I5 1.6 GHz processor. Another machine running Ubuntu 18.04 LTS with 8 GB RAM and an Intel Core I5 2.3 GHz processor served as the EHR manager (gateway). Using three database sizes: 100, 1000, and 10,000 EHR, 1, 10,100 keys fetching time-taken was measured. For each database size, Table 2 illustrates the query execution time to obtain certain keys number. It is observed that the query execution time varies depending on the retrieved keys as well as the size of the database.

The contracts are developed and implemented in a private blockchain using Ganache and Truffle. Ganache, a personal Ethereum blockchain for developers' smart contracts, construct smart contracts, decentralized applications (dApps), test software, and inspect state while maintaining control of the chain. Truffle is an Ethereum virtual machine-based programming environment, testing framework and asset pipeline for blockchains (EVM). The first thing that was noticed during the network's construction was the maximum amount of gas that each block could hold. Ganache uses 6,721,975 gas blocks by default, while Main-net presently uses 8,000,000 gas blocks. We initially confirmed the gas cost of the smart contract we designed before setting the limit quantity of gas per block on the developed blockchain. The system executes an address mapping that changes a contract's privacy preferences, as indicated in the method in Fig. 7. The contract cannot be updated if the EHR manager determines that it does not include the requestor's address. We believe that one or more research institutions may request surveillance; thus each must have its own registered address.

A smart contract uses a certain quantity of gas, which is then stored on the Ethereum blockchain. We conducted studies to determine the gas cost of each contract by adjusting the number of addresses meet the user's privacy settings. Table 3 displays the results collected. As can be seen, the cost of gas rises as the number of addresses mapped in the contract grows. We picked ten addresses for each contract since storing values on the Ethereum network has gotten expensive.

The limit amount of gas is set for each block to store 10 contracts based on the value acquired in the previous experiment. Table 4 compares contracts saved in our



```
pragma solidity 0.6.4;
contract PrivacyPreference {
    bool private preference = false;
    bool private monitoringType = false;
    mapping (address => bool) private addresses;

    constructor () public {
        addresses  [address(0x00281055afc982d96fab65b3a49cac8b878184cb16)] = true;
    }

    function changePreferences() public {
        if (addresses [msg.sender])
            preference = true;
    }

    function changeMonitoringType () public {
        if (addresses [msg.sender])
            monitoringType = true;
    }

    function preferenceStatus() public view returns (bool) {
        return preference;
    }

    function monitoringStatus() public view returns (bool) {
        return monitoringType;
    }
}
```

Fig. 7 Privacy preference smart contract

Table 3 Gas cost varying the address quantity

Addresses	Gas cost
1	183,733 gas
10	320,090 gas
100	1,524,958 gas

Table 4 Contracts stored by block in different networks

Network	Gas cost
Default Ganache	11
Ethereum Main-net	13
EHRMS Network	10

network to those stored in Main-net and the regular Ganache network. The results demonstrate that the number of contracts saved in our network with Main-net and Ganache differs by a slight margin. Because the gas size of each block in Main-net varies over time, this variation has no effect on the network's functionality.

Then, a limit of 320,090 gas per block is established in our network, allowing us to store exactly 10 contracts every block. Because we employ a private blockchain, this value has no impact on network performance.

To register and verify the contract, the registration time of contracts in the blockchain was evaluated through the gateway, taking into account that each stored EHR refers to a contract. For this experiment, we used an Ubuntu 18.04 LTS system with 2 GB of RAM and an Intel Core i7 3.8 GHz processor to host our blockchain. By establishing a link with the blockchain, we next used the web3.js package to register and validate the contracts. Following that, we measured the time it took to register 1, 10, and 100 contracts. The results are shown in Table 5. We deduced from

Table 5 Execution time for contract catalogue in blockchain

Contract	Execution time (s)
1	0.63
10	4.80
100	39.20

Table 6 Execution time to obtain a contract in blockchain

Blocks	Execution time (s)
1	0.53
10	0.57
100	0.59

these findings that time-taken to register transactions in a blockchain varies linearly with the number of contracts being registered at the same time.

Following that, the address of each contract generated was used to search the blockchain for it and analyze the gateway's connection time to the blockchain. We ignored the time it takes for the gateway to retrieve the contract address from the database. As shown in Table 6, an increase in the number of blocks has no effect on the time it takes to obtain a contract recorded on the blockchain.

5.1 Security Assessment

We will assess the security of the developed smart contract in this part. We use a methodology to identify the primary sorts of attacks that can be carried out against smart contracts in this evaluation. We discovered three plausible contract assaults among them:

1. *Reentrancy:* Calling a smart contract function many times by different users can result in inconsistencies in the function's final result. We choose to use the change Preferences method for n different users in order to evaluate this attack in our contract.
2. *Front-running:* A changePreference() transaction can be observed in the platform's mempool (i.e., memory pool) before it is processed, allowing a person to react in advance. The memory pool serves as a repository for unconfirmed transactions. A transaction is created and then sent to the network and stored in the mempool. We want to check how numerous transactions in the mempool behave in our tests.
3. *Gas limit denial of service (DoS):* When a user attempts to exceed a block's gas limit with one or more transactions, the transaction is refused, and the transaction is not executed. In our experiments, we try to find an exploit in the contract that allows us to go above the gas limit.

To carry out the first sort of attack (Reentrancy), we used multiple addresses on the blockchain to call the changePreferences() function at the same time. We did this by registering the addresses that might have access to the blockchain in the

Table 7 Reentrancy attack result varying the number of addresses

Addresses	Expected result	Final result
2	True	True
5	False	False
10	True	True

Table 8 DoS probability in relation to contract complexity

Complexity	Probability of DoS
O(1)	Impossible
O(1)	High
O(n 2)	Extremely high
O(2 n)	Extremely high
O(n!)	Extremely high

smart contract. As previously stated, the maximum number of addresses that may be stored in a contract is ten; thus the test was limited to that number. We check whether the flow of calls to a single contract function can cause inconsistencies in it in this test.

According to the results obtained, the contract established was not influenced by the Reentrancy attack, as shown in Table 7. We discovered that the attack was ineffective due to the changePreferences() function's simplicity. This assault may have an impact on contracts with a higher level of complexity.

The Front-Running attack was carried done by observing the transaction mempool in Truffle Console and using the changeMonitoringType() method. We wanted to find the blockchain transaction in the mempool after running the method. However, because we use a private blockchain with few transactions, the function was executed immediately. As a result, there was no time to complete another transaction before the previous one was completed. Another factor contributing to this effect was the contract's simplicity, which shows that when the changeMonitoringType() function was called, a new block was formed. The Bloom filter record displayed tries to protect the user's privacy and defend against third-party attacks.

We investigated the contract to generate denial of service attacks in the third test (Gas Limit DoS). We used both contract's functionalities in this experiment. This test, unlike the first, seeks to generate an exploit in the contract in order to surpass the block's gas limit. Unlike the Reentrancy attack, which aims to maliciously change the value of a transaction using an exploit without exceeding the block's gas limit, this sort of attack prevents a transaction from being executed.

Based on the results of the tests, we discovered that the contract we built is tamper-proof against Gas Limit DoS. Because of the smart contract's simplicity, when a function is called several times, it is processed promptly, preventing DoS attacks. However, we discovered that the algorithm's complexity has a direct impact on this security concern. The created algorithm's complexity is classified as O(1), making the attack unachievable. When the contract design is not done appropriately, Table 8 shows the possibility of a DoS attack occurring based on the complexity of the algorithm.

To show the contract's intricacy, we utilized the Surya tool to create a graphic representation. Surya is a smart contract system utility tool that gives a multitude of visual outputs as well as information about the contracts' structure. It also allows you to query the function call graph for manual contract inspection. The contract functions do not interact with one another and cannot be invoked externally by other smart contracts. This method simplifies contracts and eliminates security concerns.

We used the audit tool Mythril to assess the security of the constructed smart contract, as we did in the previous experiment. Mythril is an Ethereum Virtual Machine bytecode security analysis tool. It employs symbolic execution, Satisfiability Modulo Theories (SMTs) solution, and taints analysis to uncover security issues in smart contracts. This program looks for code that could lead to inconsistencies in security and can uncover flaws in Ethereum and other platforms' smart contracts.

The security warning MythX SWC-103 appears in the report generated by the Mythril program. When we specify in the contract a distinct version of the pragma utilized in the compiler, a floating pragma is issued. Version 0.6.4 was used to create the contract while version 0.6.7 was used by the solidity compiler installed on the PC used in the tests. This technique can be problematic since old versions of the pragma can cause errors in the contract's execution. To fix the problem, we'll need to update the contract to utilize the same compiler version as the machine.

5.1.1 Privacy and Sharing

Patients' privacy is preserved by separating each patient's identity from provider identities. A new Ethereum account is created for separate patient-provider relationship by health providers. This enables the patient to establish public relationships with providers without exposing personal information of the individual who are involved in the relationships. But, communication is done indirectly via their provider's account; nonetheless, the account is for patients' transactions provision with ether.

Also, the system employs content extraction signature (CES) scheme to remove sensitive part of EHR signed by the doctors. CES generates the valid signature extraction which cannot be forged without the private key of the signer.

We restrict the amount of data to be stored on the blockchain to a small set by creating the reference the EHR. The references and predefine permissions are stored on the chain. It is important that every patient should have some data storage as on-chain. A 9727-byte executed transaction creates an identity catalogue contract. Thereafter, respective patient and provider relationship requires a 5007-byte transaction, with contract updates entailing 220 bytes. This effectively provides estimation of the required bytes for storage by every single node.

5.1.2 Storage Management

Blockchains need all the nodes to store information and links it to the blockchain because blockchain cannot save massive data. If these data are stored directly in the blockchain network, it will increase computational overhead and storage burden due to the fixed and limited block size. However, data privacy leakage is imminent. We restrict the amount of data to be stored on the blockchain to a small set by creating index of the EHR, which references the HER coupled with the index that is stored on the blockchain.

We apply the architecture for off-chain storage since it enhances large storage volumes of encrypted original EHR service provider local storage, and blockchain for on-chain verification only stores few indexes of the corresponding raw data. This reduces the blockchain storage load and private data integrity and privacy. Moreover, users can leave and rejoin the system at any time, and then get access to their historical record according to the index downloaded from the most recent block in the blockchain.

5.1.3 Data Audit

EHRMS also relies on audit log management as security mechanism since some exceptions may have resulted from the misuse of access privileges or dishonest behavior by data requestors. Audit log serves as proofs when disputes arise to hold users accountable for their interactions with patient record. Thus, immutable public ledger and smart contract in the blockchain provide immutable record for all of access requests to achieve traceability and accountability. Audit log majorly consists of important information such as timestamp of logged event, requester user, data owner ID, access type (create, query, update), and validation result of the request.

6 Conclusion

Electronic health record management systems rely on sharing architectures that are state of the art to maintain privacy. The design makes use of a private blockchain, smart contracts, and CES for anonymization. Because each patient's records are kept separate, their privacy is protected. For each patient-provider connection, healthcare professionals are generating a new Ethereum account to assist individuals in establishing public relations with providers without releasing personal information. Despite the fact that communication isn't done directly through the main account of their supplier, ether is also used to provide patient transactions. The system employs content extraction signature (CES) scheme to remove the sensitive part of EHR signed by the doctors. CES generates the valid signature extraction which cannot be forged without the private key of the signer.

As a web application, the developed system can be used on a variety of platforms, including PCs and mobile devices. By allowing all network nodes to check and disseminate all transactions and block them according to the established rules defined in the smart contracts, the system strikes a balance between security and comfort.

The use of decentralization concepts and blockchain frameworks to develop robust, user-specific, and interoperable EHR systems is demonstrated in this study. Data access is managed by Ethereum smart contracts, which keep verification logs and EHR entries and enable patients with comprehensive record reviews, care auditability, and knowledge exchange through multiple storage and provider networks.

The findings validated the viability of the architecture as designed. According to the outcome of the research, the applied smart contract is impervious to a variety of attacks. With regard to security considerations, the approach utilized in this work to build a basic complexity contract supported the proposed design. To be clear, more sophisticated structures may require even more complicated contracts.

In these situations, a more thorough security study is required in order to discover any weaknesses that could jeopardize the contract's correct functioning. The research presented in this publication builds on prior findings by Yue et al. [14]. It is worthy to note that emphasis is on the established architecture's security, converging entirely on the smart contracts structure, in addition to providing more details about the proposal and evaluating its performance. As a consequence of the performance and safety testing, we are able to show that the proposed architecture can be implemented.

References

1. A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec: Using blockchain for medical data access and permission management, in *IEEE Open and Big Data (OBD) International Conference*, (2016), pp. 25–30
2. A. Ekblaw, A. Azaria, J.D. Halamka, A. Lippman, A case study for Blockchain in healthcare: "medrec" prototype for electronic health records. Medical Research Data **13**, 13 (2016)
3. F. Arlindo, S.Flavio, R. Vladimir, L.Angela, and M.Marcos, Electronic Health Records using Blockchain Technology. Future Internet for Smart Cities funded by CNPq, 2018, proc. 465446/2014-0, CAPES proc. 88887.136422/2017-00, and FAPESP, proc. 2014/50937-1
4. D. Alevtina, X. Zhigang, R. SamueL, S. Michael, W. Fusheng, Secure and trustable electronic medical records sharing using blockchain. AMIA Ann Symp Proc, 23 (2017)
5. L. Jingwei, L. Xiaolu, Y. Lin, Z. Hongli, D. Xiaojiang, G. Mohsen, BPDS: A blockchain based privacy-preserving data sharing for electronic medical records. IEEE Glob. Commun. Conf. (GLOBECOM), 22–27 (2018)
6. S. Alexaki, G. Alexandris, V. Katos, N. Petroulakis, Blockchain-based electronic patient records for regulated circular healthcare jurisdictions, in *IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*, (2018) https://ieeexplore.ieee.org/abstract/document/851495
7. A. Siyal, A. Junejo, M. Zawish, K. Ahmed, A. Khalil, G. Soursou, Applications of blockchain technology in medicine and healthcare: challenges and future perspectives. Cryptography (2019). https://doi.org/10.3390/cryptography3010003

8. L. Wanitcharakkhakul, S. Rotchanakitumnuai, Blockchain technology acceptance in electronic medical record system, in *Proceedings of the 17th International Conference on Electronic Business*, (2018), pp. 53–58
9. Q. Xia, E. Sifah, A. Smahi, S. Amofa, X. Zhang, BBDS: Blockchain-based data sharing for electronic medical records in cloud environments. Published in Information. (2017). https://doi.org/10.3390/info8020044
10. M. Holbl, M. Kompara, A. Kamišalic, A. Zlatolas, A systematic review of the use of blockchain in healthcare. In symmetry. (2018). https://doi.org/10.3390/sym10100470
11. A. Rajput, Q. Li, A. Ahvanooey, I. Masood, EACMS: Emergency access control management system for personal health record based on blockchain. IEEE Access **7**, 84304–84317 (2019)
12. D. Nguyen, P. Pathirana, M. Ding, A. Eneviratne, Blockchain for secure EHRs sharing of mobile cloud based E-Health systems. IEEE Access **7**, 66792–66806 (2019)
13. R. Guo, H. Shi, Q. Zhao, D. Zheng, Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. IEEE Access **6**, 11676–11686 (2017)
14. Y. Xiao, W. Huiju, J. Dawei, L. Mingqiang, J. Wei, Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. J. Med. Syst. **40**, 218 (2016). https://doi.org/10.1007/s10916-016-0574-6
15. K. J. Peterson, R. Deeduvanu, P. Kanjamala, & K. Mayo, "A blockchain-based approach to health information exchange networks", 2016.
16. Y. Huihui, and Y. Bian, "A blockchain-based approach to the secure sharing of healthcare data" 2017.
17. A.J. Zargar, M. Manzoor, T. Mukhtar, Encryption/decryption using elliptical curve cryptography. Int. J. Adv. Res. Comp. Sci. **8**(7) (2017)