High-Confidence Computing 3 (2023) 100101

Contents lists available at ScienceDirect

High-Confidence Computing

journal homepage: www.sciencedirect.com/journal/high-confidence-computing

Research

d-EMR: Secure and distributed Electronic Medical Record management

Ehab Zaghloul, Tongtong Li, Jian Ren *

Department of Electrical and Computer Engineering, Michigan State University, USA

ARTICLE INFO

Article history:
Received 5 October 2022
Revised 3 December 2022
Accepted 3 December 2022

Keywords:
Privilege-based access
Hierarchy
Attribute-based encryption
EMRs
Distributed data sharing
Blockchain
Smart contract

ABSTRACT

As more and more data is produced, finding a secure and efficient data access structure has become a major research issue. The centralized systems used by medical institutions for the management and transfer of Electronic Medical Records (EMRs) can be vulnerable to security and privacy threats, often lack interoperability, and give patients limited or no access to their own EMRs. In this paper, we first propose a privilege-based data access structure and incorporates it into an attribute-based encryption mechanism to handle the management and sharing of big data sets. Our proposed privilege-based data access structure makes managing healthcare records using mobile healthcare devices efficient and feasible for large numbers of users. We then propose a novel distributed multilevel EMR (*d*-EMR) management scheme, which uses blockchain to address security concerns and enables selective sharing of medical records among staff members that belong to different levels of a hierarchical institution. We deploy smart contracts on Ethereum blockchain and utilize a distributed storage system to alleviate the dependence on the record-generating institutions to manage and share patient records. To preserve privacy of patient records, our smart contract is designed to allow patients to verify attributes prior to granting access rights. We provide extensive security, privacy, and evaluation analyses to show that our proposed scheme is both efficient and practical.

1. Introduction

It was reported that during 2021, over 45 million people experienced healthcare data breach incidences [1] with the cost soared from an average $7.13 million in 2020 to $9.3 million per occurrence, which is a 29.5 percent increase [2]. To mitigate financial loss and implications on the reputation associated with data breaches, large organizations, such as healthcare networks, government agencies, banking institutions, commercial enterprises, etc., began allocating resources into data security research to develop and improve accessibility and storage of highly sensitive data.

Record-sharing between medical institutions can help improve medical diagnoses, treatment decisions, and enhance the overall patient experience. Patients generally must rely on the record-generating institution to do the sharing in an efficient, secure, and private way. This incurs additional computational costs to securely store, maintain, and transmit records when requested by other institutions. In the case of a nation with a central healthcare network, the systems employed may be mismanaged, antiquated, and potentially jeopardize patient data. Therefore, there has been an effort to develop comprehensive, secure, and private electronic record sharing systems that also eliminate reliance on the record-generating institution.

In the U.S., the Health Insurance Portability and Accountability Act of 1996 (HIPAA) [3] sets a guideline for secure and private use of patient health data. It does not define, however, how these systems are to be developed and applied. As a result, many centralized and private applications have come to market in the U.S. as ways to manage EMRs. Because these systems are often built on proprietary technology, lack of interoperability between medical institutions is a major concern. Meaning that, in majority, the burden of transferring health records lies on patients, oftentimes in printed copies from one medical institution to another. In other cases, patient records are transferred via fax or systems similar to electronic mail. These common transfer methods can be inefficient and possibly jeopardize patient privacy and data security. There is also a lack of an auditable trail of data transfer, and there is no way to preclude who will view a patient record on the other end of a fax. Lastly, in cases where there is no likely method of data transfer or data cannot be trusted, physicians may resort to duplicate testing causing resource waste.

Other nations with universal healthcare may use a national system, such as the National Health Service (NHS) [4] in the United Kingdom. While the NHS provides patients some control over how their data is shared with the ability to opt-out of the Summary Care Record (SCR) using an SCR opt-out form. The use of a centralized system presents security and privacy complications. Depending on the administrative practices of the General Practitioner, action in response to the form may take days to weeks to

* Corresponding author.
E-mail address: renjian@msu.edu (J. Ren).

https://doi.org/10.1016/j.hcc.2022.100101

complete. In addition, since the NHS is a central entity compared to individual medical institutions in the US system, a malicious actor could potentially access more patient EMRs if the system becomes compromised.

In this paper, a privilege-based data access structure is first proposed to solve the problems of sharing data within organizations with complex hierarchies. Then we propose a distributed multilevel EMR (*d*-EMR) management scheme that leverages blockchain and smart contracts to eliminate the record-generating institution from the record-sharing process. One goal is to eliminate the need of a trusted third party, such as banks, to facilitate payments between individuals. Shortly after the advent of Bitcoin, based on the concept initially introduced by Nick Szabo in 1994 [5], smart contracts were incorporated into blockchain-based systems. Smart contracts are self-executing pieces of code that can digitally enforce and facilitate verified negotiations of contracts between two participating individuals without the need of a trusted third party. They continue to appear in today's systems such as Ethereum [6], which aim to provide services beyond simple payments. Using such smart contracts, users can design and customize their own systems on top of blockchains such as Ethereum.

In the existing systems, EMRs are stored within the generating institution, which significantly limits the patients' access to their own records. They must also trust that the institutions will not leak any sensitive information of their EMRs to unintended organizations. In comparison, our proposed work empowers them over their EMRs. First, it provides record ownership to the patients, allowing them to be in actual possession of their records, rather than have them stored by the generating institution. Second, it allows patients to selectively share different parts of their records with distinct data users based on their privacy preferences.

The work presented in this paper expands significantly on the model presented in [7]. In this model, patients have no method to monitor access to their EMRs. They must rely on trusted third parties to efficiently share this information. In comparison, our proposed scheme is distributed in terms of data management and storage, and grants patients control over their records, thereby minimizing the dependency on trusted third parties. Patients can also independently trace the history of access to their EMRs. The main contributions presented in this paper can be summarized as follows:

(1) We present multiple data file partitioning techniques and propose a privilege-based data access structure that facilitate data sharing in hierarchical settings.
(2) We develop a novel distributed electronic medical record-sharing scheme for patients that is secure, preserves the privacy of patient data, and is more efficient than traditional paper record sharing.
(3) We propose a distributed method for verifying medical institution staff member attributes over the blockchain before issuing them access keys. This method can help minimize blind reliance on key-issuers whose behavior cannot be verified when validating attributes of staff members.
(4) We conduct comprehensive security and privacy analyses of the proposed scheme.
(5) We implement our proposed scheme using smart contracts and deploy them over the Ethereum blockchain for performance evaluation and empirical results.

In Section 2, the related work is reviewed. In Section 3, preliminaries are introduced that summarize key concepts used in this research. Next, in Section 4, the problem formulation is described, outlining the system model and our design goals. In Section 5, our proposed scheme is presented in detail, outlining the proposed algorithms. Following that, in Section 6 we present security analysis of our proposed schemes. In Section 7, we conduct a performance and application analysis that is supported with numerical results. Finally, in Section 8, a conclusion is drawn to summarize our research.

2. Related work

Hierarchical Attribute-Based Encryption (HABE) that combines the Hierarchical Identity-Based Encryption (HIBE) [8] and Ciphertext Policy Attribute-Based Encryption (CP-ABE) [9] was introduced in [10,11]. HABE is able to achieve fine-grained access control in a hierarchical organization. File Hierarchy CP-ABE (FH-CP-ABE) [12] is one of the most recent hierarchical solutions available today. It proposes a leveled access structure to manage a hierarchical organization that shares data of various sensitivity. A single access structure was proposed that represents both the hierarchy and the access policies of an organization.

A decentralized data management scheme was introduced in [13] that facilitated access-control management over a blockchain. In this system, the actual data records are stored in an off-blockchain storage while pointers to these records are maintained by a key–value storage over the blockchain. This solution helps simplify the amount of data processed on the blockchain. However, the method used to define access policies in this scheme does not consider hierarchical data sharing. A user that desires to share files selectively among multiple users in a hierarchy will have to define several access policies that could become a complex problem as the number of users increases drastically.

MedRec [14], the first functional electronic medical record-sharing system built on some concepts from [13] was introduced. This work builds on three Ethereum [6] smart contracts that manage the authentication, confidentiality, and accountability during the data sharing process. In this system, the primary entities involved in maintaining the blockchain are the parties interested in gaining data, such as researchers and public health authorities. In return, the institutions are rewarded with access to aggregate and anonymized data. However, the success of such a system is dependent on the participation of entities that maintain the system in return for data. Similar to [13], MedRec does not consider hierarchical data sharing.

Another functioning electronic medical record-sharing scheme was presented in [15] to provide a secure solution using blockchain. However, the system uses a cloud-based storage system to store medical records. With centralized storage, the system becomes liable to a single point of failure. Moreover, similar to MedRec, this work builds over a private and monitored blockchain where each node involved in maintaining consensus is known. In comparison, our proposed work eliminates the need for centralized storage by relying on distributed storage such as IPFS. It also builds over a permissionless blockchain instead of a private one, mitigating the reliance on predefined validating nodes in a private blockchain setting. Moreover, in contrast to both schemes, we also present a security analysis and show that our proposed solution is secure against potential attacks.

Most recently, some research considered using the distributed ledger directly for electronic health records storage, which is not only very costly but also unfit for large amount of data storage [16].

3. Preliminaries

3.1. Access structure

An access structure defines sets of attributes that a single individual is entitled to accessing the protected data. Let $\{P_1, \ldots, P_n\}$ be a set of parties. A set of parties that can reconstruct the secret is defined as a collection. The collection is monotone meaning that, if $\mathscr{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ then $\forall B \in \mathscr{A}$ and $B \subseteq C$ implies $C \in \mathscr{A}$. An access structure is a monotone collection $\mathscr{A}$ of non-empty subsets $\{P_1, \ldots, P_n\}$, i.e., $\mathscr{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \emptyset$. The sets in $\mathscr{A}$ are called the authorized sets and the sets not in $\mathscr{A}$ are called the unauthorized sets.

In this paper, we use parties to represent the attributes. This means that an access structure $\mathscr{A}$ may consist of both authorized and unauthorized sets of attributes.

3.2. Ciphertext policy attribute-based encryption

Ciphertext Policy Attribute-Based Encryption (CP-ABE) is a fine-grained access control and encryption scheme. It allows users to share their data selectively by using access policies that are integrated into the ciphertexts during encryption. CP-ABE formally divides the process into four main functions.

(i) $\mathsf{Setup}(1^\kappa)$: a probabilistic function with input security parameter κ. The function is performed by a key-issuer to generate a public key PK and the master key MK.

(ii) $\mathsf{KeyGen}(\mathsf{MK}, \mathbb{A})$: a probabilistic function carried out by a key-issuer to generate a unique secret key SK for a data user based on the MK and a set of attributes $\mathbb{A} = \{\mathsf{A}_1, \mathsf{A}_2 \ldots, \mathsf{A}_n\}$ possessed by the user.

(iii) $\mathsf{CPABE\text{-}Enc}(\mathsf{PK}, \mathsf{m}, \mathcal{T})$: a probabilistic function carried out by the data owner to encrypt message m under an access policy $\mathcal{T}$ and generate ciphertext CT.

(iv) $\mathsf{CPABE\text{-}Dec}(\mathsf{CT}, \mathsf{SK})$: a deterministic function carried out by the data user to decrypt a ciphertext CT using the uniquely generated secret key SK for the user.

3.3. Blockchain

A blockchain is a public ledger that stores all the cryptographically processed transactions performed over a peer-to-peer (P2P) network. For this paper, we refer to the open-source Ethereum blockchain, which provides its users with a platform to build, deploy and run decentralized applications [17]. The P2P network nodes offer a decentralized Turing-complete virtual machine named the Ethereum Virtual Machine (EVM). It can execute smart contracts deployed by the users over the public and non-trusted network nodes. In addition, log entries may be contained in receipts which are attached to transactions executed by the EVM. These logs represent the results of *events* fired from the smart contracts.

3.4. InterPlanetary file system

InterPlanetary File System (IPFS) [18] is a peer-to-peer (P2P) network used to store and share content-addressable data over the network nodes in a distributed manner. The network runs a stack of protocols that are responsible for storing and accessing the data stored over it.

3.4.1. Identities

To join the P2P network, nodes first generate unique node identities before establishing connections.

Table 1
Notations summary.

Symbol	Definition
p_a	The ath patient in the set of patients $\mathcal{P}$
m_b	The bth medical institution in the set of institutions $\mathcal{M}$
$s_{b,l}$	The lth staff member working at m_b
pr	Private key of $s_{b,l}$
pub	Public key of $s_{b,l}$
$\mathbb{A}$	The set of attributes of a staff member
$\mathcal{T}_i$	The ith access policy defined by patient at level $\mathcal{L}_i$
$\mathcal{L}_i$	ith level within H where $1 \leq i \leq k$
sk_i	The ith secret key belonging to $\mathcal{T}_i$
SBK	Subscription-based key
R_i	The ith partition of record $\mathcal{R}$
ER_i	The ith encrypted record partition under $\mathsf{sk}_i \parallel \mathsf{SBK}$
PK	Public key of key-issuer
MK	Master key of key-issuer
Esk_i	Encryption of sk_i under PK
SK	Secret key issued for a staff member
ESK	Encryption of SK under pub

3.4.2. Network

The network layer in IPFS manages the established connections between the connected peers.

3.4.3. Routing

The routing layer keeps track of the addresses of nodes within the network and the data they store. It uses a Distributed Hash Table (DHT) to identify the data stored over any node of the network.

3.4.4. Exchange

IPFS uses BitSwap, a data exchange protocol inspired by BitTorrent [19], to exchange data between nodes.

3.4.5. Objects

IPFS partitions data files into segments and photographically hashes each segment, where the hashes are used to reference the location of the corresponding segments (see Table 1).

4. Problem formulation

Patient record is a data file that contains items of different security requirements and can be chosen to share with a wide spectrum of distinct individuals that may include but is not limited to admittance staff members, physicians assisted by nurses and technicians that provide the required treatment, and maybe even financial or discharging staff members. Selectively sharing data files on the cloud becomes a burden on the data owner as the hierarchy grows (the access privileges increase in number) and/or as the access restrictions become more complex due to an increase in the sensitivity of the file segments.

If those staff members can access the necessary parts of the patient medical record(s), it can expedite their specific jobs, such as patient admittance and treatment, and prevent iatrogenic illnesses caused by administering inappropriate medication or harmful drugs. However, for the sake of patient privacy, staff members should only be given permission to access patient record(s) from the medical history based on the profile of the patient.

A trivial solution is to encrypt each part of the patient record using public-key encryption and grant access to the private keys to the staff members based on their permissions. However, this solution is quite inefficient due the large number of encryptions and storage spaces required. Therefore, the challenge is to provide the data owners with an efficient, secure and privilege-based method that allows them to selectively share their data files

among multiple data users while minimizing the required cloud storage space needed to store the encrypted data segments.

In current systems, records often have a variety of attributes and may be blocked or intentionally delayed by competitive record-generating institutions. To prevent this, patient should have full control over their medical records and determines sensitivity of their own record attributes. The challenge is how to provide patients with a method that could allow them to share the attribute values of their records efficiently and securely while maintaining the privacy preferences they desire. Patients should also be able to selectively share certain parts of their record(s) with the healthcare providers they select. Although patients may not easily understand how to share the medical parts of their record(s), empowering them would at least allow them to consult their trusted medical staff such as their Primary Care Physicians (PCPs) to decide on how and which parts of their medical record(s) are to be shared. With their consent, patients may even delegate this process to such entities to avoid situations such as being unconscious, requiring someone to make decisions for them. Lastly, patients should not be concerned with the availability nor the guaranteed secure storage of their records.

4.1. System model

The general model of our proposed record-sharing scheme is illustrated in Fig. 1. The system consists of six main entities.

4.1.1. Patients

Patients are data owners that wish to share their data selectively based on their privacy preferences. We denote the set of patients as $\{p_a \in \mathcal{P} \mid 1 \leq a \leq \infty\}$.

4.1.2. Medical institutions

Medical institutions provide treatment and generate medical records for the patients. We denote the set of medical institutions as $\{m_b \in \mathcal{M} \mid 1 \leq b \leq \infty\}$.

4.1.3. Staff members

Personnel employed at a medical institution. They are the data users. We denote the set of staff members at medical institution m_b as $\{s_{b,l} \mid \forall\ 1 \leq l \leq \infty\}$. Staff members are categorized into groups of similar characteristics based on the set of attributes $\mathbb{A} = \{\mathsf{A}_1, \mathsf{A}_2 \ldots, \mathsf{A}_n\}$ they each possess. Attributes represent identifiers that are selected from an infinite pool set. They may be as general as the role of a staff member and could be as unique as a biometric.

4.1.4. Key-issuer

A semi-trusted party that generates keys for permissioned staff members upon their requests to access parts of the record.

4.1.5. Distributed storage P2P network

A non-trusted P2P network used by the patients to store and share their records with staff members at any medical institution.

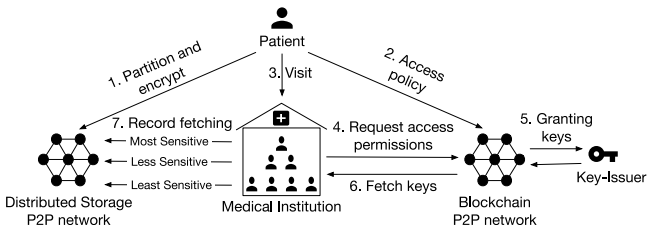

Fig. 1. General scheme orchestration.

4.1.6. Blockchain P2P network

A non-trusted P2P network that maintains a blockchain to regulate access permissions of patient records to the staff members of medical institutions.

In general, in order for patients, medical institutions, staff members or key-issuers to interact with any of the two P2P networks, they must run nodes that are capable of connecting to either network. These nodes may be light-weight nodes (similar to the simple payment verification nodes in Bitcoin [20]) that can assemble transactions and propagate them to the network. Running light-weight clients does not require powerful computing machines and can be done via simple machines such as mobile devices making our proposed scheme accessible to everyone. The common requirement for running either node is being able to generate the public key pub and private key pr pair.

4.2. Design goals

Our proposed scheme has the following design goals:

4.2.1. Data ownership

Patients are empowered over their records and control how to share them.

4.2.2. Fine-grained access control

Patients can grant specific access permissions to the desired staff members based on their privacy preferences. They can selectively share any part of their records using any set of staff member attributes they wish, limiting access to specific parts of their record and to certain staff members at particular medical institutions.

4.2.3. Collusion resistant

Two or more staff members that possess different sets of attributes and/or are employed at the same/different institution(s) cannot combine their attributes to gain access to any part of the records they are not authorized to access individually.

4.2.4. Data confidentiality

Records are completely protected from any unauthorized staff members that do not possess the correct set of attributes predefined by p_a. This includes any type of space that stores the records for the patients.

4.2.5. Distributed storage

Patient records are stored in a distributed storage network. They can be accessed at any time and by any permissioned staff member. Storage is not centralized and/or controlled by the record-generating institution.

4.2.6. Data access traceability

Patients can trace back to the entire history of access to their records. This allows patients to keep track of how their records are being accessed.

5. The proposed distributed EMR management scheme

In this section, we first present an overview of the major chain of events in our proposed scheme. For discussion purposes, we assume that a patient has already received some medical treatment at institution m_b and that a medical record $\mathcal{R}$ was generated for him/her. We also assume that p_a anticipates visiting some other institution in the future and would like to share $\mathcal{R}$ selectively among its staff members. Based on our description, we then propose a generalized structure through three smart contracts that can be deployed on a blockchain. We assume patients and data users interact with our smart contracts via easy-to-use and user-friendly interfaces, hence, onboarding new users to our proposed model becomes a trivial task.

5.1. Privilege-based access structure

Fig. 2 illustrates the general privilege-based data access structure. Data users are ranked into k levels of privileges, $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$. The data owner defines an access tree $\mathcal{T}_i$ at each corresponding level $\mathcal{L}_i$. Each $\mathcal{T}_i$ is associated with the appropriate leaf nodes (attributes) that define the privileges of the level. A data user that possesses the attribute satisfies $\mathcal{T}_i$ at $\mathcal{L}_i$ is granted access to symmetric key sk_i and derive $\{\mathsf{sk}_{i+1}, \ldots, \mathsf{sk}_k\}$ from sk_i using a one-way cryptographic function h as follows:

$$\mathsf{sk}_{i+1} = h(\mathsf{sk}_i). \tag{1}$$

Therefore, the data owner only needs to select secret key sk_1.

As shown in Fig. 2, an access tree $\mathcal{T}_i$ may consist of non-leaf nodes and leaf nodes. The non-leaf nodes are threshold gates, represented as G, while the leaf nodes are attributes, represented as A. The DO may construct access trees from any number and layout of nodes that satisfy the privacy preferences desired.

One of the advantages of our proposed privilege-based data access structure is the ability to reduce attribute replication when defining the hierarchy. Data users that possess attributes which can satisfy access tree $\mathcal{T}_i$ are granted access to sk_i, however, they do not need to possess attributes that can also satisfy the access trees $\{\mathcal{T}_{i+1}, \ldots, \mathcal{T}_k\}$ in order to obtain $\{\mathsf{sk}_{i+1}, \ldots, \mathsf{sk}_k\}$. This helps simplify the process of defining a hierarchy as the number of users and/or access constraints grow. With reduced-size access trees, we can greatly reduce the computational complexity when encrypting file partitions, generating private keys for privileged users and decrypting ciphertexts.

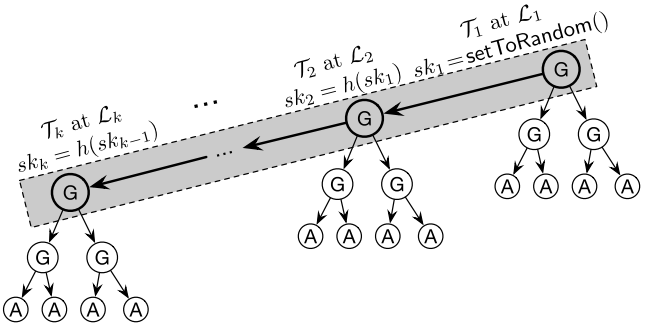

Fig. 2. Privilege-based multilevel access structure.

5.2. Privilege-based record partition and sharing

The data owner partitions the medical record $\mathcal{R}$ into a set of k data sections based on the record attributes, that is $\mathcal{R} = \{\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_k\}$. Each $\mathsf{R}_i \in \mathcal{R}$ is treated as a new file that is associated with a sensitivity value used to assign access rights to the data users based on their privileges. The process of partitioning $\mathcal{R}$ is performed based on the privilege-based access structure of $\mathcal{R}$. We assume that $\mathcal{R}$ consists of at least one record, resulting in multiple ways to partition it as shown in Fig. 3.

If $\mathcal{R}$ consists of a single record, then each $R_i \in \mathcal{R}$ represents one or more record attribute(s) associated with the record, as shown in case (1) in Fig. 3. However, if $\mathcal{R}$ consists of multiple records, then the data owner has flexibility in choosing how to partition it. One approach is to handle each record as a whole, where records are clustered into groups of similar sensitivity. In this case, each $\mathsf{R}_i \in \mathcal{R}$ represents one or more record(s), as shown case (2) in Fig. 3. Alternatively, partitioning can be performed over specific record attributes, versus the whole record. In this case, $\mathsf{R}_i \in \mathcal{R}$ represents one or more record attribute(s) of the records, as shown case (3) in Fig. 3.

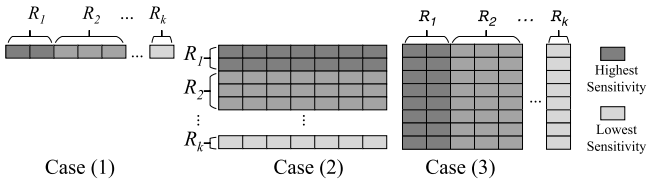

Fig. 3. File partitioning.

Regardless of how partitioning is performed, suppose R_1 contains the most sensitive information of $\mathcal{R}$ that can only be accessed by a data user at the highest level $\mathcal{L}_1$ and R_k contains the least sensitive information of $\mathcal{R}$ that can be accessed by all data users at any level of the hierarchy. Before the data owner uploads $\{\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_k\}$ to the cloud, each $\mathsf{R}_i \in \mathcal{R}$ is encrypted separately as follows:

$$\mathsf{ER}_i = \mathsf{Enc}_{\mathsf{sk}_i \| \mathsf{SBK}}(\mathsf{R}_i), \tag{2}$$

where Enc is a symmetric encryption algorithm such as the Advanced Encryption Algorithm (AES) and SBK denotes the subscription-based key. The SBK can only be accessed by either a system call or an attributed-based protection to active members through successful membership subscription authentication. Members have no direct access to SBK and can only use it while on-site. This design serves two purposes: (i) it prevents any users from sharing the ER_i's even if they share their sk_i's with others and give them unprivileged access and (ii) it provides an easy option to disable unsubscribed members from accessing EMRs. In other words, with this design, key revocation is no longer needed. Finally, using the Encryption function discussed in Section 3.2, p_a encrypts each sk_i under its corresponding $\mathcal{T}_i$ defined in the privilege-based access structure, such that

$$\mathsf{Esk}_i = \mathsf{CPABE\text{-}Enc}(\mathsf{PK}, \mathsf{sk}_i, \mathcal{T}_i), \tag{3}$$

where CPABE-Enc is the CP-ABE encryption function and PK is the public key generated during the Setup phase. p_a then stores the generated ciphertexts $\{\mathsf{ER}_1, \mathsf{ER}_2, \ldots, \mathsf{ER}_k\}$ and $\{\mathsf{Esk}_1, \mathsf{Esk}_2, \ldots, \mathsf{Esk}_k\}$ over IPFS.

In cases that EMRs consist of multiple/diverse attributes, it may be a burden for patients to define their privilege-based access structures and may even leak sensitive information if defined inappropriately. A possible approach that patients may consider is to divide their records into multiple categories based on the nature of the data being shared. For example, a record could be divided into personal information, medical information, and financial information. For each category, a patient can then define a separate privilege-based access structure which would reduce the overall complexity of each structure. Patients may even consult their trusted Primary Care Physicians (PCPs) on how to define a privilege-based access structure for proper medical information sharing.

5.3. Access policy

To facilitate data management and sharing, p_a shares the privilege-based access structure that incorporates the access policies $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$. The access structure is incorporated into a smart contract that is then deployed over the blockchain.

Medical institution visit. When p_a visits a medical institution $\{\mathsf{m}_b \mid b \geq 1\}$ to get medical treatment, p_a interacts with various staff members that may or may not belong to a predefined privilege-based access structure.

Requesting access permissions. Permissioned staff members will be granted access to certain parts of a record based on the attributes they possess. To request access, a staff member $s_{b,l}$ interacts with the smart contract deployed by p_a that incorporates the privilege-based access structure. The smart contract will verify whether the $s_{b,l}$ possesses the attributes that can satisfy an access policy within the access structure. The smart contract will publicly announce the granted access permissions of the $s_{b,l}$ over the blockchain.

Granting keys. The key-issuer continuously monitors the blockchain for access announcements. An announcement contains the set of attributes $\mathbb{A}$ that the $s_{b,l}$ possesses. It is a form of verification to the key-issuer that the $s_{b,l}$ possesses before generating a secret key SK using the KeyGeneration function described in Section 3.2. The key-issuer then encrypts SK with the public key pub of $s_{b,l}$ as

$$\mathsf{ESK} = \mathsf{Asym\text{-}Enc}_{\mathsf{pub}}(\mathsf{SK}), \tag{4}$$

where Asym-Enc is the asymmetric encryption function. To share the key with the $s_{b,l}$, the key-issuer transacts with a global smart contract that publicly announces the encrypted key over the blockchain.

Key fetching. The $s_{b,l}$ can obtain the uniquely encrypted secret key ESK by monitoring the announcements made over the blockchain. Once obtained, the $s_{b,l}$ can decrypt ESK using his/her private key pr that corresponds to the public key pub such that

$$\mathsf{SK} = \mathsf{Asym\text{-}Dec}_{\mathsf{pr}}(\mathsf{ESK}), \tag{5}$$

where Asym-Dec is the asymmetric decryption function.

Record fetching. The storage location of the encrypted EMRs over IPFS is also provided to the $s_{b,l}$ in an encrypted form under his/her public key in the announcement made over the blockchain. The $s_{b,l}$ decrypts the IPFS location to fetch the EMRs. Here, the $s_{b,l}$ possesses the necessary key to decrypt the parts he/she has been granted access to. Once fetched, the $s_{b,l}$ can decrypt the encrypted symmetric key Esk_i using the derived key SK as explained in Section 3.2. That is

$$\mathsf{sk}_i = \mathsf{CPABE\text{-}Dec}(\mathsf{Esk}_i, \mathsf{SK}), \tag{6}$$

where CPABE-Dec is the CP-ABE decryption function. After obtaining sk_i, the $s_{b,l}$ can derive the remaining set of secret keys $\{\mathsf{sk}_{i+1}, \ldots, \mathsf{sk}_k\}$ using Eq. (1). Finally, the $s_{b,l}$ can decrypt each encrypted partition, such that

$$\mathsf{R}_i = \mathsf{Dec}_{\mathsf{sk}_i \| \mathsf{SBK}}(\mathsf{ER}_i), \tag{7}$$

where Dec is the decryption function.

5.4. Smart contracts

The proposed scheme includes three main smart contracts.

*Access member registration (*AMR*).* AMR is a global smart contract that registers staff members of medical institutions for patient EMRs access. The process of registering staff members by interacting with this smart contract can be delegated to a number of certified institutions that verify staff members against their possessed attributes before registering them. This is achieved by incorporating precise policies into the smart contract which requires certain identities to trigger it. Once a certified institution verifies the attributes of a staff member, it executes the smart contract and uses the attributes as input. This results in the attributes are stored over the blockchain. This process maps the identity (in our case the Ethereum address) of the staff member to the set of attributes possessed. By observing the blockchain, we can trace back the ongoing activity of these certified institutions as they add, delete, or modify the information of staff members, hence, we can also detect malicious data manipulation.

Algorithm 1: AMR Smart Contract

```
1  struct staffMember contains
2      staffID
3      staffAttributes[upBound]
4  end
5  variable mapping(address → staffMember) Map
   // Adding a new staff member
6  function
     addStaffMember(_address, _id, _attributes[upBound])
7      if (msg.sender ∉ setOfCertifiers) then
8          throw
9      else
10         Map(_address → _id, _attributes[upBound])
11     end
12 end
   // Return the attributes of a staff member
13 function getAttributes(_address)
14     return Map[_address].staffAttributes
15 end
```

Algorithm 2: AVPA Smart Contract

```
1  Initialized variables _address = AMR address
2  event LogAnnounce(_address, attributes, T_i)
3  function verifyRequest(_msg, σ)
       // Validate digital signature
4      signer ← ecrecover(_msg, σ)
5      if (signer ≠ Hash(pub)) then
6          throw
7      else
           // Fetch stored attributes of the staff member
8          r ← AMR(_address)
            fetched ← r.getAttributes(signer)
           // Fetched attr. vs access policies
9          for i ← 1 to k do
10             if (satisfy(T_i, fetched) == true) then
11                 LogAnnounce(signer, fetched, T_i)
12                 break
13             end
14         end
15     end
16 end
```

Algorithm 1 outlines the general functions of the AMR smart contract. It includes two parts: (i) the data structure staffMember that the smart contract uses to store new staff members. It incorporates the identity staffID and the array of attributes staffAttributes of the staff member, and (ii) the two main functions addStaffMember and getAttributes of the smart contract. The addStaffMember function allows only certified institutions setOfCertifiers to upload the data of new staff members after physically verifying their attributes. On the contrary, the getAttributes function can be called by any user.

*Access verification and permission announcements (*AVPA*).* AVPA, outlined in Algorithm 2, is a unique smart contract defined by each patient. It incorporates his/her personally defined privilege-based access structure to facilitate selective record management

Algorithm 3: GK Smart Contract

```
1 event LogKeys(staffAddress, ipfsAddress)
  // Sharing the location of a generated access key
2 function addKey(_staffAddress, _ipfsStorageAddress)
3   if (msg.sender ∉ setOfIssuers) then
4     throw
5   else
6     LogKeys(_staffAddress, _ipfsStorageAddress)
7   end
8 end
```

and sharing. The staff members interested in obtaining parts of the record interact with AVPA contracts to request access permissions.

AVPA performs a sequential verification process represented in a single function verifyRequest. It takes two inputs: a message _msg and its signature σ. That is

$$\_msg = \mathsf{Hash}(\mathsf{staffID}), \tag{8}$$

$$\sigma = \mathsf{Sign}(\mathsf{pr}, \mathsf{pub}, \_msg), \tag{9}$$

where Hash is the hashing function and Sign is an ECDSA signing function.

In the initial verification process, the AVPA smart contract uses an ECDSA compatible validation function ecrecover(_msg, σ) to validate σ by verifying whether

$$\mathsf{ecrecover}(\_msg, \sigma) = \mathsf{Hash}(\mathsf{pub}) \tag{10}$$

holds true. Next, the signer is compared to the address of the requesting staff member as shown in line 5. If the addresses match, the contract uses the signer to fetch the previously verified and stored attributes of this staff member in the AMR contract. However, this method is liable to replay attacks. Our countermeasure is discussed in Section 6.

If the initial verification is successful, AVPA checks fetched attributes against the access policy $\mathcal{T}_i$ within the access structure as outlined in lines 10–14. The access structure is defined by p_A during contract deployment and maintains the desired privacy settings as discussed in Eqs. (2) and (3). The access policies are tested sequentially starting from the highest-ranked access policy $\mathcal{T}_1$ at level $\mathcal{L}_1$. If the attributes satisfy the access policy $\mathcal{T}_i$, the verification is discontinued and the function fires an event LogAnnounce that announces a permanent log of the smart contract transaction stored over the blockchain. This event is a public and immutable announcement to ensure that the staff member in possession of signer should be granted access to the parts of the record $\{\mathsf{R}_i \dots R_k\}$ corresponding to levels $\{\mathcal{L}_i \dots \mathcal{L}_k\}$.

*Granting keys (*GK*).* GK, outlined in Algorithm 3, is a global smart contract that is used to share the location of the generated and encrypted access keys over the distributed storage. GK generally consists of a single function.

When the key-issuer observes a LogAnnounce event fired by the AVPA smart contract, it generates an access secret key SK for the specified staff member using the unique set of attributes fetched stored in the logs. Next, the key-issuer encrypts this access key as shown in Eq. (4) with the public key of the staff member. It also encrypts the IPFS record location as (ERA = $\mathsf{Asym\text{-}Enc}_{\mathsf{pub}}$(_ipfsRecordAddress) with the same public key and uploads both values to the IPFS storage, maintaining their reference location. Following that, the smart contract fires an event, LogKeys as shown in line 6, which permanently stores the address of the requesting staff member _staffAddress along with the IPFS storage location _ipfsStorageAddress. Using _ipfsStorageAddress, the staff member can fetch the encrypted key ESK and ERA stored over the network. However, as shown in the addKey function, only certified key-issuers setOfIssuers are capable of triggering this function, which can prevent attackers from spamming the smart contract logs with fake keys or addresses. The staff member can listen for these fired events and obtain the corresponding _ipfsStorageAddress as it appears in the logs and use it to fetch ESK and ERA from the network. It is important to note that only the staff member in possession of the private key pr corresponding to the public key pub will be able to decrypt the fetched encrypted key and IPFS record address. First the staff member decrypts _ipfsRecordAddress = $\mathsf{Dec}_{\mathsf{pr}}$(ERA) to obtain the location of the encrypted record. Next, using the secret key SK, the staff member decrypts Esk_i to generate the secret key sk_i. Finally, the staff member decrypts ER_i stored over IPFS at _ipfsRecordAddress to obtain the record part R_i. Attackers continuously listening to the fired events may be able to learn certain IPFS addresses storing generated encrypted keys, but not the actual keys or the record address.

5.5. Access permission revocation

Attributes possessed by members may change over time due to, for example, job switch or retirement. As a result, the access rights to p_a records should be revoked to be consistent with the access policy. However, this could be challenging since it requires the attributes of the staff members to be updated periodically. While adding an expiration date [21] to each attribute when registering staff members seems to be a simple solution, it requires the patients to incorporate time constraints into their access policies.

The proposed distributed multilevel EMR management scheme can handle access permission revocation efficiently without requiring any extra process for the key-issuer. The staff members only need to be registered to ensure attribute-based access control. As a result, if at any point in time a staff member is unable to provide evidence of registration to the level of access claimed, future access to the patient records will be disabled. Consequently, if the staff member attempts to request records he/she is no longer entitled to access, the AVPA smart contract will fail to verify the request.

For data users that have already accessed certain records and are no longer registered, our proposed scheme can also revoke their access permissions since each record partition R_i is encrypted under $\mathsf{sk}_i \parallel \mathsf{SBK}$. To revoke access from those data users that are no longer registered, p_a only needs to generate a new subscription-based key SBK then re-encrypt his/her record partitions under $\mathsf{sk}_i \parallel \mathsf{SBK}$. This design does not require regenerating a new set of keys or making any changes to the privilege-based access structure. To prevent any data user that has been granted key sk_i at some point with an inactive membership from accessing the record partitions, we only need to re-encrypt the record partitions either periodically (such as monthly), or based on demand.

6. Security and privacy analysis

In this section, we present the security and privacy discussions of our proposed scheme. We assume that a symmetric encryption technique such as AES is used to secure each individual data file R_i and the process of attribute authentication between a patient and the key-issuer, in order for the data user to obtain a private key, is secure and efficient. We also assumed that our system runs over a blockchain that is maintained by a significant number of nodes, for example, Ethereum. In such blockchain platforms, it is very expensive to tamper with verified transactions processed into

blocks unless the attackers can control more than half of the computational power in the network. We also consider distributed storage networks such as IPFS that are content addressable. These networks use the hash computation of the original data when storing and referencing it over the network nodes, resulting in tamper-resistant storage. Moreover, we consider adversaries that can act as an entity within the system and can manage to connect to either the IPFS or the blockchain network. Such adversaries are capable of generating fraudulent transactions and propagating them through the blockchain network. By fraudulent transactions, we mean transacting with smart contracts in an effort to access data to which they are not granted access. They may also deploy their own smart contracts over the blockchain, request/store data over the IPFS nodes, and continuously monitor the network channels.

6.1. Security analysis

Following the work presented in [7,22], we have the two theorems for the proposed privilege-based access structure.

Theorem 1. *The proposed privilege-based access structure is secure against unprivileged accesses assuming the hash function is collision-resistant.*

The security of the proposed privilege-based access structure is reduced to the hardness of the DBDH problem. Based on Theorem 1 and by proving that a single esk_i at any $\mathcal{L}_i$ is secure, the whole system is proved to be secure since all ciphertexts at any level follow the same rules.

Theorem 2. *The proposed privilege-based access structure is secure against adaptively chosen-plaintext attacks if the DBDH assumption holds.*

Theorems 3 and 4 prove that the proposed EMR is secure against replay and DDoS attacks. Theorem 5 shows that the EMS smart contracts are collusion resistant. Theorem 6 evinces that the EMR storage is tamper-resistant. The detailed proofs of these results can be found in [23].

Theorem 3. *The proposed distributed EMR management scheme is secure against replay attacks.*

Theorem 4. *The proposed distributed EMR management scheme can counter Distributed Denial of Service (DDoS) attempts.*

Theorem 5. *The proposed smart contracts of the distributed EMR management are collusion resistant.*

Theorem 6. *Using a content-addressable storage such as IPFS ensures that data is tamper-resistant.*

6.2. Privacy analysis

While our proposed scheme aims at satisfying the privacy desires of patients and their records, it tends to leak knowledge about the staff members and their access patterns. Each staff member must initially become registered and is linked to a unique address before engaging with the system. Since this information can be leaked, attackers can simply monitor requests triggered by the staff members by observing the blockchain activity. However, since no information is revealed about the records of patients through the AVPA smart contracts, attackers can only track when staff members trigger requests, but not the data they have requested or the patient information.

From the perspective of patients, our proposed scheme is intentionally designed to allow them to trace back the accesses performed by staff members to their records. This is possible by tracing the history of LogAnnounce events fired as outlined in line 12 of Algorithm 2. All fired events are permanently stored over the blockchain, allowing the patients to continuously monitor how their records are being accessed. If a patient notices irregular staff member accesses that are undesired, the patient can simply redeploy a new AVPA smart contract that defines new or more strict access policies.

Theorem 7. *Under the proposed model, the privacy of patient records location is preserved.*

Proof. First, since the user record is encrypted as described in Eq. (4) before it is uploaded to IPFS, the content of the user record is protected. Second, for the current IPFS system, any user in possession of the data can reproduce its cryptographic hash, search its corresponding location that is maintained by the DHT, and locate it within the peer-to-peer network nodes determined by the default IPFS partitioning techniques. This will also result in recursively locating any data linked to it. To ensure data privacy, we will append a random value r to the record before uploading it to IPFS. Therefore, the storage location is determined by

$$h = \mathsf{Hash}(\mathcal{R} \parallel r). \tag{11}$$

The randomness of r makes it infeasible for the attacker to locate the data stored over IPFS nodes. □

7. Performance analysis and evaluation

7.1. Computational cost of privilege-based access structure

We formulate the encryption and decryption costs based on the number of bilinear mapping operations involved in the Encrypt and the Decrypt functions for each scheme.

7.1.1. Encryption cost

Encryption cost is measured as the number of basic operations involved in generating the ciphertext from the plaintext. It is formulated based on the Encrypt function. The privilege-based access structure generates a ciphertext for each level of the hierarchy. The number of operations involved is $(2(|Y_1|+\cdots+|Y_k|)+k)$ and $2k$ respectively, where $|Y_1|, |Y_2|, \ldots, |Y_k|$ are the number of leaf nodes (attributes) associated with access trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ respectively.

Compared with hierarchical schemes such as HABE and FH-CP-ABE, the privilege-based access structure minimizes the overall number of operations. This is because the encryption process for schemes such as HABE and FH-CP-ABE involves more complex hierarchies and access trees that contain all the access policies for all the levels. Moreover, privilege-based access structure involves smaller access trees, each one limited to level-specific attributes and policies.

7.1.2. Decryption cost

When comparing privilege-based access structure to HABE, the number of operations involved are similar. For both schemes, encryption complexity would greatly depend on the number of attributes possessed by the user and the access tree generated during encryption.

7.2. Smart contracts computational complexities

AMR *Contract.* As outlined in Algorithm 1, the addStaffMember function takes _attributes as input to register a new staff member. Due to the computational limitations of the EVM, returning dynamic content from external function calls is not possible, i.e. returning a dynamically sized array of attributes resulting from the getAttributes function when called by the AVPA contract. This means the size of Map[_address].staffAttributes must be fixed in order to be executed by the EVM. As a result, the only workaround is to use statically-sized upBound arrays of attributes as input to the addStaffMember. In this case, performance is based on the size of upBound, such that

$$\text{Performance} \propto \text{upBound}. \tag{12}$$

AVPA *Contract.* As discussed in Algorithm 2, the AVPA contract consists of two sequential verifications that play a dominant role in the performance of the contract. In the initial verification, an input digital signature σ is validated to prove the identity of the requesting staff member. With Solidity, the only available cryptographic function that can perform such a process is the ECDSA ecrecover function. The function recovers the Ethereum address associated with the public key from the elliptic curve signature or returns zero on error. The ecrecover function is considered to be relatively expensive. At the time of writing, the ecrecover function requires 3000 gas units. However, this initial verification is fixed with each AVPA contract transaction.

Assuming the initial verification is successful, the AVPA externally fetches the attributes of the staff member to test them against the incorporated access structure. Similar to the AMR contract, the performance of the AVPA contract is dependent on the upBound of the fetched attributes. Finally, the fetched attributes are tested against the access policies starting at the highest level within the hierarchy. Here, performance is affected by the complexity of the overall access structure defined and the number of levels k within the hierarchy. Assuming the worst-case scenario, a requesting staff member might have his/her attributes tested against all access policies within the access structure and only receive the least sensitive part R_k of the record or nothing at all. The computational complexity can be represented as $\mathcal{O}(k \times \text{upBound} \times |\mathcal{T}_i|)$, where $|\mathcal{T}_i|$ is the number of attributes that form the access policy at level $\mathcal{L}_i$. To reduce this complexity, we can modify the AVPA contract such that the staff members can request that their attributes be tested against only specific access policies rather than being tested sequentially against all policies. This would reduce the computational complexity to $\mathcal{O}(\text{upBound} \times |\mathcal{T}_i|)$. We can also incorporate optimized search functions, for example, binary search, that can help optimize searching for attributes in the fetched set against those in the access policies. This means that we can further reduce the computational complexity to $\mathcal{O}(\text{upBound} \times \log |\mathcal{T}_i|)$. However, it is important to note that in such a scenario, the patients need to make their access structures publicly available in order for the staff members to request certain access policies. This results in a trade-off between the performance of the AVPA contract and the privacy of the access policy defined.

GK *Contract.* The GK contract requires the least computational complexity and gas costs when compared to the AMR and AVPA smart contracts. The computations involved are as simple as firing a LogKeys event when access permissions are granted to staff members. The gas costs of such operations are minimal in comparison to the other computations in the AMR and AVPA smart contracts.

7.3. Smart contracts gas costs

In this section, we implement, simulate and present the estimated costs required to deploy/transact with the smart contracts of our proposed scheme in multiple scenarios. Our experiments are performed over the Ethereum testnet blockchain [24]. We specifically choose the Ethereum blockchain given that it is by far the most widely used smart contract hosting public blockchain. We intend to show that the costs of running distributed multi-level EMR management over a public blockchain are reasonable and that our results may even be improved when choosing a consortium or private blockchain. We also note that our proposed scheme can be implemented over any other blockchain that incorporates smart contracts.

In our simulation, the AVPA smart contract with the highest degree of complexity is implemented. When executed, the AVPA smart contracts sequentially check whether the attributes of the requesting users satisfy the access policies in order starting from the highest level of the hierarchy. As discussed in Section 7, there is a trade-off between privacy and performance. Therefore, we note that our presented results can be further enhanced in terms of performance as discussed previously. However, we intentionally choose to implement our smart contracts this way to show that even with these conditions, our proposed contracts are still cheaper when compared to the current systems used by the record-generating institutions to share medical records. In contrast to the current record-sharing systems, our proposed scheme eliminates the role of the record-generating institution completely. This results in eliminating other overhead costs required when sharing medical records. For example, in developed countries such as the U.S., record-generating institutions must comply with certain state laws that enforce a maximum fee a record-generating institution may charge when requested to share its records. Given the current competitiveness, in most cases, the medical institutions would charge the entire allowed fees to maximize their profits.

For our simulations, we select the number of levels in the hierarchy to be $k = \{5, 10\}$, the total number of attributes used in the entire access structure to be $N = \{25, 50, 100\}$, and $\text{upBound} = \{10, 20, 30, 40, 50\}$. Without loss of generality, we also assume that the number of attributes for each level follows the uniform distribution. For example, if $k = 5$ and $N = 50$, then the number of attributes used to define an access policy is $|\mathcal{T}_i| = 10$. The following experiments have been conducted through an Ethereum node running on a system with 1.8 GHz Intel Core i5 and 8 GB RAM. Our numerical results are the averages of 10 trials under each scenario.

Based on our experiments, the costs of deploying our smart contracts are not affected by the value of upBound since there is no major change in code as this value changes. Therefore, for all values of upBound that we tested, the costs remained constant.

Table 2 summarizes these costs in terms of gas limits and their estimated and equivalent costs in ETH and United States Dollar (USD) at the time of testing. To measure the optimum gas limit for each smart contract deployment, we used the JSON-RPC method, eth_estimateGas [25], that generates and returns an estimate of the required gas limit based on the network success rate. We also set the gas price to 20 GWEI, where 1 ETH = 1×10^9 GWEI. We intentionally select this value relatively higher than the average gas prices specified in [26] for guaranteed and fast processing. Again, we note that our presented results can be further optimized by selecting reduced gas price values. As demonstrated in Table 2, the estimated costs for deploying the AMR and GK smart contracts remain constant as we alter the values of k and N (highlighted in Table 2). However, the costs for deploying the AVPA smart contracts increase with an increase in the values k and/or N.

Table 2
Estimated smart contract deployment costs.

Smart contract	$k = 5$, $N = 25$				$k = 5$, $N = 50$				$k = 10$, $N = 100$			
	Gas Limit	Cost/ETH	Cost/USD	Data/bytes	Gas Limit	Cost/ETH	Cost/USD	Data/bytes	Gas Limit	Cost/ETH	Cost/USD	Data/bytes
AMR	240383	0.004809	1.42	736	240447	0.004809	1.42	736	240447	0.004809	1.42	736
AVPA	857419	0.017148	5.21	3323	1132628	0.022653	6.67	4536	1303607	0.026072	7.56	5188
GK	125617	0.002512	0.74	304	125617	0.002512	0.74	304	125617	0.002512	0.74	304

Table 3
Estimated costs to run the AVPA smart contract.

upBound	$k = 5$ $N = 25$ $\mathcal{L}_1$	$k = 5$ $N = 25$ $\mathcal{L}_5$	$k = 5$ $N = 50$ $\mathcal{L}_1$	$k = 5$ $N = 50$ $\mathcal{L}_5$	$k = 10$ $N = 100$ $\mathcal{L}_{10}$	$k = 5$ $N = 100$ $\mathcal{L}_{10}$
10	3.66	7.96	6.73	14.63	6.40	22.29
20	3.70	8.22	6.77	14.95	6.40	22.59
30	3.76	8.54	6.86	15.31	6.42	23.18
40	3.86	8.86	7.00	15.67	6.45	23.43
50	3.92	9.12	7.08	16.14	6.55	23.96

Table 4
Estimated addStaffMember function costs.

	upBound				
	10	20	30	40	50
Gas limit	246531	449890	653249	856674	1060034
Cost/ETH	0.004931	0.008998	0.013065	0.017133	0.021201
Cost/USD	1.45	2.65	3.85	5.06	6.29

Table 5
Estimated addKey function costs.

	upBound				
	10	20	30	40	50
Gas limit	26379	26380	26375	26379	26378
Cost/ETH	0.000528	0.000528	0.00053	0.000524	0.000528
Cost/USD	0.14	0.14	0.14	0.15	0.15

Table 3 demonstrates the changes in USD costs of transacting with already deployed AVPA contracts as we alter the values k and N for permissioned staff members at levels $\mathcal{L}_1$, $\mathcal{L}_5$ and $\mathcal{L}_{10}$. In Table 4 we present the cost for the addStaffMember, which increases as the value upBound increases, while in Table 5, we present the cost for addKey function, which remains constant regardless of the upBound value used.

7.4. Symmetric encryption overhead

Our proposed work does not require patients to reconstruct their privilege-based access trees and regenerate a new set of keys. In order to revoke access from certain data users, patients need only to symmetrically re-encrypt their EMRs under the same set of keys $\{sk_1, \ldots, sk_k\}$ along with a new SBK. The overhead for re-encrypting EMRs can be very low when utilizing a high encryption/decryption implementation such as AES, Counter Mode (CTR) operations, and Cipher-Block-Chaining (CBC). According to the Crypto++ 5.6.0 Benchmarks,[1] the performance for AES/CTR on an Intel Core 2 1.83 GHz processor measures under Windows Vista are 139 MiB/Second, 113 MiB/Second, and 96 MiB/Second for 128-bit, 192-bit, and 256-bit key sizes respectively. Similarly, the performance measures for AES/CBC are 109 MiB/Second, 92 MiB/Second, and 80 MiB/Second for 128-bit, 192-bit, and 256-bit key sizes respectively. These commonly used cryptographic algorithms are widely accepted and incur minimal costs making our model feasible.

[1] https://www.cryptopp.com/benchmarks.html

7.5. Extended application discussions

Aggregated patient data has the capability of transforming the future standard of care for patients through the application of predictive analytics and big data methods. One of the biggest challenges to using health data to its fullest extent is the inaccessibility and segmentation of EMRs [27]. As demonstrated by our analyses, the proposed scheme can eliminate these constraints. It grants healthcare providers the ability to efficiently extract knowledge from disconnected EMRs that have been willingly shared. For example, a meaningful opportunity for big data analytics in this context is the capability to model drug and treatment efficacy. Healthcare providers can access previously shared EMRs to understand how life-saving drugs and treatments perform, respective to the disease state and profile of the patient. This data can be used to make enhanced and customized treatment decisions for patients. As healthcare treatment becomes more individualized, successful patient outcomes are more likely and a one-size-fits-all approach to patient treatment is no longer adequate.

8. Conclusion

In this paper, we proposed a Privilege-based access structure allows data to be shared efficiently and securely on the cloud. Based on this, we proposed distributed multilevel EMR management scheme, a secure, privacy-preserving and distributed data sharing scheme that runs over a blockchain using smart contracts. We demonstrated that distributed multilevel EMR management can be used in cases such as medical record-sharing where patients require an efficient and selective method to share their records with staff members of medical institutions. The proposed scheme eliminates the reliance on the record-generating institutions when data is shared and completely empowers the patients. Our security and privacy analyses show that the proposed scheme is secure and preserves the privacy of patient records. We also presented some recommended security practices developers should keep in mind when developing their systems. Finally, our comprehensive evaluation proves the efficiency of the proposed scheme and presents numerical results that support our performance analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (CCF1919154, ECCS-1923409).

References

[1] H. Journal, Healthcare data breach statistics. [Online]. Available: https://www.hipaajournal.com/healthcare-data-breach-statistics/.

[2] C. Group, Cost of healthcare data breach average balloons to $9.3 million, 2021.

[3] US Dept of Health and Human Services, Heath insurance portability and accountability act, 2018, [Online]. Available: https://www.hhs.gov/hipaa.

[4] N. England, Nhs England and nhs improvement, 2019, [Online]. Available: https://www.england.nhs.uk.

[5] N. Szabo, Smart contracts: building blocks for digital markets, EXTROPY J. Transhumanist Thought (16) (1996).

[6] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper, Vol. 151, 2014, pp. 1–32.

[7] E. Zaghloul, T. Li, J. Ren, An attribute-based distributed data sharing scheme, in: 2018 IEEE Global Communications, IEEE, 2018.

[8] C. Gentry, A. Silverberg, Hierarchical id-based cryptography, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2002, pp. 548–566.

[9] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy, SP'07, IEEE, 2007, pp. 321–334.

[10] G. Wang, Q. Liu, J. Wu, Hierarchical attribute-based encryption for fine-grained access control in cloud storage services, in: Proceedings of the 17th ACM Conference on Computer and Communications Security, ACM, 2010, pp. 735–737.

[11] G. Wang, Q. Liu, J. Wu, M. Guo, Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers, Comput. Secur. 30 (5) (2011) 320–331.

[12] S. Wang, J. Zhou, J.K. Liu, J. Yu, J. Chen, W. Xie, An efficient file hierarchy attribute-based encryption scheme in cloud computing, IEEE Trans. Inf. Forensics Secur. 11 (6) (2016) 1265–1277.

[13] G. Zyskind, O. Nathan et al, Decentralizing privacy: Using blockchain to protect personal data, in: Security and Privacy Workshops, SPW, IEEE, 2015, pp. 180–184, 2015 IEEE.

[14] A. Azaria, A. Ekblaw, T. Vieira, A. Lippman, Medrec: Using blockchain for medical data access and permission management, in: Open and Big Data, OBD, IEEE, 2016, pp. 25–30.

[15] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, F. Wang, Secure and trustable electronic medical records sharing using blockchain, in: AMIA Annual Symposium Proceedings, 2017, American Medical Informatics Association, 2017, p. 650.

[16] U. Chelladurai, S. Pandian, A novel blockchain based electronic health record automation system for healthcare, J. Amb. Intell. Hum. Comput. 13 (2022) 693–703, [Online]. Available: http://dx.doi.org/10.1007/s12652-021-03163-3.

[17] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform, 2013, 2017, [Online]. Available: http://ethereum.org/ethereum.html.

[18] J. Benet, IPFS - content addressed, versioned, P2P file system (draft 3). [Online]. Available: https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf.

[19] Bittorrent, 2018, [Online]. Available: http://www.bittorrent.com.

[20] S. Nakamoto, Bitcoin: A p2p electronic cash system, 2008, [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[21] M. Pirretti, P. Traynor, P. McDaniel, B. Waters, Secure attribute-based systems, J. Comput. Secur. 18 (5) (2010) 799–837.

[22] E. Zaghloul, K. Zhou, J. Ren, P-MOD: Secure privilege-based multilevel organizational data-sharing in cloud computing, IEEE Trans. Big Data 6 (4) (2020) 804–815.

[23] E. Zaghloul, T. Li, M. Mutka, J. Ren, *d*-MABE: Atributed multilevel attribute-based EMR management and applications, IEEE Trans. Serv. Comput. 15 (3) (2022) 1592–1605.

[24] Ropsten (revival) testnet, 2018, [Online]. Available: https://ropsten.etherscan.io.

[25] Json RPC, 2018, [Online]. Available: https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_estimategas.

[26] Ethereum average gas price chart, 2018, [Online]. Available: https://etherscan.io/chart/gasprice.

[27] S.E. White, A review of big data in health care: challenges and opportunities, Open Access Bioinform. 6 (2014) 13–18.