Hindawi
Security and Communication Networks
Volume 2022, Article ID 4941214, 13 pages
https://doi.org/10.1155/2022/4941214

Research Article

A Blockchain-Based Personal Health Record System for Emergency Situation

Yuan Liu,[1] **Yan Du**,[2,3] **Yanan Zhang**,[2,3] **Yuan Li**,[4] **Leung Cyril**,[5] **Chunyan Miao**,[6] **Qingfeng Tan**,[1] **and Zhihong Tian**[1]

[1]*Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, Guangdong 510006, China*
[2]*Software College, Northeastern University, Shenyang, Liaoning 110169, China*
[3]*China-Singapore International Joint Research Institute, Guangzhou, Guangdong 510663, China*
[4]*Qi Lu Hospital of Shandong University, Jinan, Shandong 250012, China*
[5]*Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly, Nanyang Technological University, Singapore*
[6]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*

Correspondence should be addressed to Yuan Li; liyuan91700@163.com and Qingfeng Tan; tqf528@gzhu.edu.cn

Received 23 July 2022; Accepted 20 September 2022; Published 8 October 2022

Academic Editor: Yinbin Miao

A personal health record (PHR) system stores personal health-related information, which can assist physicians in quickly forming appropriate treatment plans in emergency situations. Because a PHR contains lots of sensitive information, the patients are only willing to share their records with authorized doctors with their permission. There are three main challenging issues: (1) it is costly to store and maintain the growing PHRs data; (2) the existing PHR systems still face the privacy leakage risk during data transmission and access control processes; and (3) the response speed cannot meet the need in an emergency situation, especially when the patients are unconscious. In this paper, based on the permissioned blockchain Hyperledger Fabric, we propose a PHR management system preserving patients' privacy and also supporting emergency access. In the system, we use reencryption technology and the anonymous identity mapping mechanism to protect patient privacy and use smart contracts to define access control strategies in an emergency situation. Furthermore, we use a quick response code and bloom filter to optimize system performance. The security analysis and experimental results show that our proposed framework guarantees the privacy of patient data from multiple aspects while improving the efficiency with which doctors can obtain PHR information.

1. Introduction

A personal health record (PHR) including healthcare history, medical records, allergy history, and genetic diseases is an important property of each patient [1]. Patients own their PHRs, and they do not have to disclose all their records to doctors whenever they seek medical treatment, unless in the case of major diseases. Meanwhile, doctors should also be allowed to check their patients' PHRs, even in emergency situations, when the patients may be unconscious and their lives are in danger [2].

Many systems have been designed to manage and access patients' personal health records based on traditional centralized databases [3, 4] or blockchain [5]. When a patient seeks treatment from a doctor, the doctor can view the patient's personal health information under the patient's permission. However, these systems still have several challenging problems, especially in an emergency situation. Firstly, its storage is costly for a healthcare service institute or hospital in recording and maintaining the fast-growing volume of personal information. In order to overcome the storage cost, the health recording service may only promise to maintain the records happening in the recent 3 years or 5 years and discard the older records. Secondly, the health records are stored in a centralized server and the data is at risk to be leaked when the data is transmitted or accessed by doctors. Since health records contain highly private and sensitive information, data leakage can result in exposure to privacy. Thirdly, the traditional access control methods bear relatively long time delay without considering the emergency requirements; meanwhile, the quick access mechanism is credential for patients.

Aiming at solving the above three problems, this paper designs and implements a personal health record system based on blockchain. The system is composed of three main modules: personal health records storage, emergency access management, and encrypted transmission. The increasing data of all patients are stored in the distributed file system IPFS [6, 7] and, to avoid a vast amount of data occupying massive blockchain space, only the hash of the uploaded personal health record file on IPFS is stored in the blockchain. In addition, the access control strategy is set on the blockchain and executed by smart contracts automatically. While the unconscious patient with certain identifying information is under treatment, smart contracts check the identity of the emergency doctor who applies for the PHR file according to the access control list (ACL). ACL determines whether the emergency physician is allowed to access the data by screening the doctor's ID. An application is launched after obtaining the information about the patient (the information is not the patient's personal health record information, but the information used for identity authentication, while letting the system know which patient's information to obtain). This information is described as a quick response code in this paper.

The most important part of this system is the protection of patients' personal health records. The protection consists of three parts: anonymous storage of encrypted private keys, the encryption of personal health record files, and the encryption of message transmission. The anonymous identity mapping mechanism [8] makes it impossible for anyone other than the patient to know the encrypted file corresponding to the private key, so, even if someone gets the private key, they do not know which file to decrypt. This part ensures the security of the smart contract execution that decrypts the private key. The second part ensures the privacy and security of the personal health record file. The personal health record file is encrypted by the key from AES algorithm [9] and it is encrypted again by the public key of patient (notice that the key from the AES algorithm is different from the public key and private key of patient). Encrypted files are stored on IPFS that returns hash values to the patient. Furthermore, the patient's private key is encrypted and then stored on the blockchain (the private key is encrypted and anonymized to prevent leakage, and it is decrypted during the execution of the smart contract). The third part guarantees the security of message transmission during the application of data. When an authorized doctor applies the patient's data, the smart contract will send the corresponding hash value of the patient's encrypted personal health record file. The doctor can ask for the file from IPFS through the given hash. After obtaining the double-encryption file, the doctor will apply for the decryption of the patient's private key to completely decrypt the original PHR file encrypted from the AES algorithm and the patient's public key. What is more, the smart contract is launched to reencrypt the decrypted AES key by the doctor's public key and return it to the doctor. The doctor can then decrypt it with his/her private key and see the patient's original PHR file.

Meanwhile, the design of this system also considers the response time to race against time for the patient's life. Each patient can generate a quick response code that includes his/her basic information, representing his/her identity. Doctors enter the system by scanning codes to save the time of finding and inputting patient information. Furthermore, the bloom filter [10] is used to store the access control list, which can quickly filter the medical doctor's identification to accelerate the authentication of the doctor's identity.

Our system is a Hyperledger Fabric [11] based blockchain application that realizes data sharing with emergency doctors while ensuring the privacy of patients' personal health records in emergency situations. In addition, according to experimental verification, the performance of our proposed system is greatly optimized, which can effectively accelerate the speed of doctors obtaining a patient's PHR. The contributions of this paper are as follows:

(i) We achieve doctors' access without the authorization of the patient's supervision in an emergency situation through escrowing the patient's encrypted private key on the blockchain and setting the access control list on the smart contract.

(ii) We use symmetric encryption, asymmetric encryption, and reencryption technologies to protect the privacy of patients' PHR and introduce the anonymous identity mapping mechanism to protect the encrypted private key escrowed on the blockchain.

(iii) In order to optimize the system, we introduce the quick response code and bloom filter to accelerate emergency physicians to obtain the patient's PHR data and also design a smart contract to verify that PHR data has only been added but not changed when updating.

The remainder of this article is organized as follows. Section 2 introduces some related work. Section 3 describes the proposed model. Section 4 analyzes the security and privacy of the proposed model. Section 5 presents the results of the simulation experiments and evaluates the performance of the proposed model. Finally, the paper is summarized in Section 6.

2. Related Work

In this section, we summarize some outstanding research work currently solving problems in the PHR system, while discussing the weakness of existing solutions.

In order to better store and share the patient's personal health records, researchers introduced the semitrusted server to implement data storage and proxy reencryption. Li et al. [17] proposed a framework for access control to PHR stored in a semitrusted server, using attribute-based encryption (ABE) technology to encrypt the PHR file of each patient. Users in the PHR system are divided into multiple security domains, which reduces the complexity of key management for users and ensures a high degree of privacy for patients. Bhatia et al. [18] proposed a lightweight certificateless proxy reencryption scheme to make the PHR system capable of low-power mobile devices, which uses the semitrusted proxy server to perform the reencryption process. Reference [14] proposed a revocable and unpaired ciphertext policy attribute-based encryption for the management of personal health records and added a proxy decryption server to decrypt the partial ciphertext at the decryption end, which can effectively reduce the computational overhead of the decryption end. Although the above models protect the privacy of the PHR system and consider efficiency issues, their designs all rely on the semitrusted server. The server is extremely vulnerable to single-point attacks, which cannot guarantee the security of data.

After discovering the shortcomings of the centralized PHR system, a lot of research work introduced blockchain. Hussien et al. [16] proposed an attribute searchable encryption method based on the smart contract to achieve secure and fine-grained access control. They introduced distributed storage IPFS to avoid storing large amounts of data on the blockchain and used one-to-many encryption to prevent unauthorized users from accessing data stored in IPFS. Wang et al. [19] proposed a new data integrity verifiable PHR sharing scheme based on blockchain. The new scheme uses searchable symmetric encryption and attribute-based encryption technology to achieve fine-grained access control without the involvement of a third party. Thwin and Vasupongayya [20] proposed a blockchain-based PHR model, which uses proxy reencryption and access control list to achieve flexible access control to the patient's PHR, revokes the doctor's access authority by updating the access control list, and uses blockchain to record access logs to ensure data auditability and nontampering. Although the above PHR systems use blockchain technology, they are designed only for conscious patients, and the patient can authorize when the doctor requests access to the data. The emergency situation is not considered, such as how to handle data requests from doctors while the patient is unconscious.

In dealing with emergency access, Huda et al. [12] introduced a new privacy-aware protocol for handling healthcare professionals' access to patient-controlled PHR in emergency situations. It uses an IC card embedded with a patient's emergency access report for strong authentication. The method of storing data on the IC card is risky. If the IC card is lost, it will cause irreparable losses. The emergency access policy designed in [15] is to preset a timeout period. If the patient rejects the emergency request, the patient is conscious and the doctor needs to be authorized to access normally. If the patient does not operate over time, the system will approve an urgent request. The setting of the timeout period here is a problem. Reference [21] proposed three verification methods for the PHR system in emergency situations. The first is that the telecommunication provider determines whether the patient's telecommunication equipment is within a reasonable distance of the hospital location. If reasonable, the telecommunication provider provides the patient's key sharing. The second is to send a shared key request to emergency contacts if the patient does not respond within a reasonable time. The third is to combine the first and second methods; if the patient's emergency contacts do not answer the phone, the first method will be used. References [22, 23] and [24] used the second method but cannot guarantee that emergency contacts respond to data requests rapidly. Reference [25] proposed a context-aware technology to realize automatic transmission of authorization in the PHR system. According to different roles, the permissions granted are different. Some role owners, such as emergency physicians, obtain additional permissions for certain data objects, thus ensuring access to PHR data when the patient is unconscious. Reference [13] also used a role-based authorization method. Although the duration of emergency doctor access is limited, it does not achieve complete privacy protection, because the patient's PHR information is not private to the emergency doctor regardless of whether the patient is in an emergency. Through the analysis, it can be seen that current access control strategies in the PHR systems under emergency situations have not achieved the desired effect.

Compare our system with the system proposed in the existing papers based on the above analysis from five aspects: decentralized, patient data privacy, access control, considering emergency, and accelerating response. The comparison results are shown in Table 1. It can be seen from the comparison results that part of the systems proposed in the existing papers does not consider solving the problems that exist in emergency situations and, currently, there are few considerations for optimizing the PHR system and accelerating the response speed of the system.

3. The Proposed System Model

In this section, we will introduce our system model in detail, including system architecture, system workflow, and system optimization.

3.1. System Architecture.

The architecture of the system model proposed in this paper is shown in Figure 1. Firstly, the patient provides his/her basic information and personal health records. The complete PHR file is encrypted by symmetric encryption algorithm AES and asymmetric encryption algorithm RSA. Then, the encrypted data is uploaded to the distributed storage network IPFS, and the hash value returned by IPFS is recorded on the blockchain. The encrypted private key of the patient is also stored on the blockchain, because of considering the patient's unconscious situation. Here, in order to ensure the security of the encrypted private key, this paper introduces the anonymous identity mapping mechanism to store the encrypted private key on the blockchain. In addition, each patient generates a quick response code (QR) based on his/her basic information to indicate his/her identity. The doctor scans the QR code to get the patients' basic information and quickly enters the system. Smart contracts deployed on the blockchain authenticate the identity of the doctor who enters. The authorized doctor initially obtains the encrypted data. This

TABLE 1: The comparsion of different PHR system.

PHR system name	Decentralized	Patient data privacy	Access control	Considering emergency	Accelerating response
Huda et al. [12]	✗	✓	✓	✓	✗
Rajupt et al. [13]	✓	✗	✓	✓	✗
Liu and Xu [14]	✗	✓	✓	✗	✗
Son et al. [15]	✓	✓	✓	✓	✗
Hussien et al. [16]	✓	✓	✓	✗	✗
Our system	✓	✓	✓	✓	✓

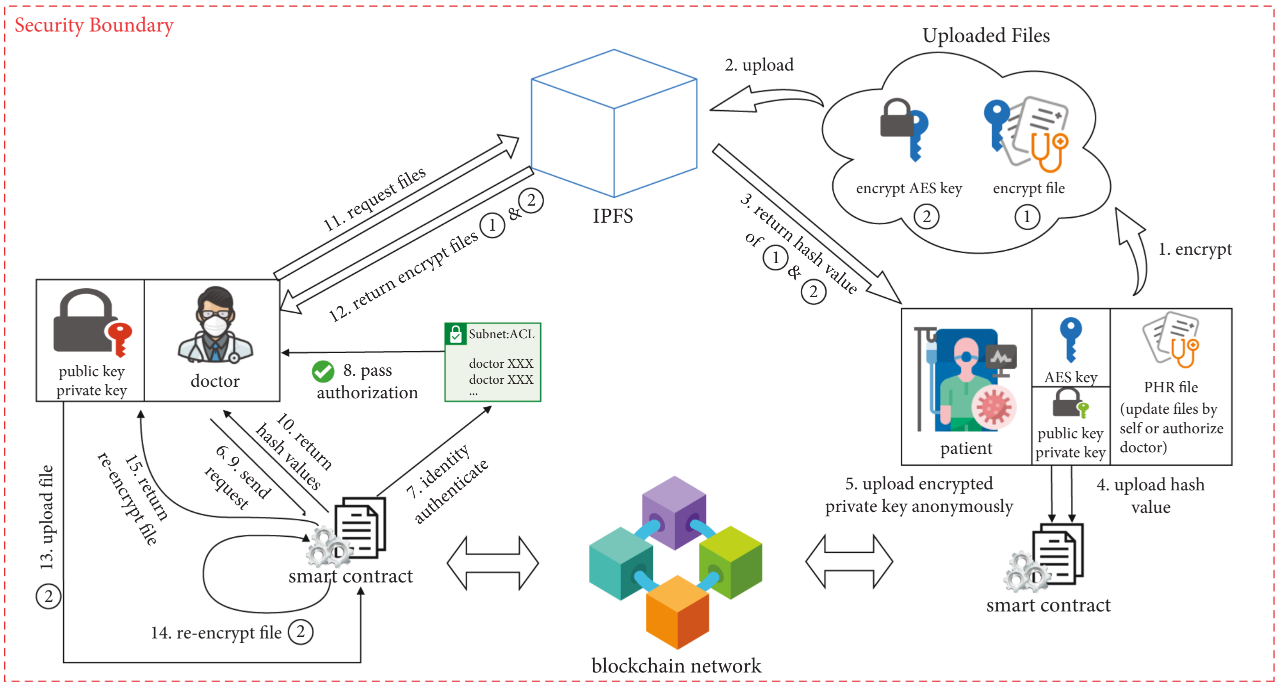

FIGURE 1: The proposed system architecture.

data is first reencrypted by the smart contract and then decrypted by the doctor. For the update of the PHR data, the patient can update the data himself/herself, and the doctor can also help the patient update the medical record with the patient's permission. In the update, a smart contract is designed to verify that the data has only been added and not changed. Our system contains the following five entities:

(1) patient is an entity that owns PHR data and wishes to share his/her PHR data with the doctor when necessary. The patient has absolute control over PHR data; he can define access control strategies and decide who can access his/her data. Considering that data must be accessed in an emergency situation, the patient escrows the encrypted private key on the blockchain and provides the corresponding decryption method in the smart contract. Besides, every patient has a quick response code that represents his/her identity.

(2) doctor is an entity that applies for access to PHR data and accurately treats patients based on existing data. The doctor quickly enters the system by scanning the QR code to authenticate the identity and then queries the patient's hash values stored on the blockchain based on the information in the QR code. The doctor downloads the corresponding encrypted file on IPFS according to the hash values and then sends the file to the smart contract for reencryption. The doctor finally decrypts the reencrypted file with his/her own private key to get the complete PHR data.

(3) IPFS is an entity used to store encrypted files. It is a distributed file system based on content addressing. After patients upload encrypted files, they will get the corresponding hash values, and the hash value of each file is unique.

(4) Blockchain network is responsible for storing some hash values and recording access logs. The blockchain in this system refers to Hyperledger Fabric, which only allows partial organizations to access it.

(5) Smart contract is an agreement that can automatically perform some functions without the intervention of a third party. The smart contract automatically performs operations such as decryption of the patient's private key, re-encryption of PHR data, and verification of data only being added without change.

Table 2: Explanation of notations in different transaction processes.

Notation	Description
PID	The ID represents patient's identity
DID	The ID represents doctor's identity
$\text{AnonymityID}_{\text{pid}}$	The patient's anonymous identity
HNumber	The number of doctors' hospitals
pk_{pid}, sk_{pid}	The patient's public key and private key
pk_{did}, sk_{did}	The doctor's public key and private key
Num	Length of the generated symmetric key
K	AES symmetric key
C	Original PHR file
C'	Updated PHR file
C_1	File with symmetric encrypted PHR
C_1'	Updated symmetric encrypted PHR file
C_2'	File with encrypted K
C_3'	File with reencrypted EnK
Ensk_{pid}	The patient's cyclically encrypted private key
EnK	K's ciphertext after being encrypted by PK_A
EnK$'$	EnK's reencrypted key
Hash_1	Hash value returned by IPFS uploading file C_1
Hash_2	Hash value returned by IPFS uploading file C_2
Hash_3	Hash value returned by IPFS uploading file C_1'

3.2. System Workflow. In this part, we will introduce the main workflow of the system from the perspective of the patient. First, the patient stores the PHR data. Then, the doctor requests to access the patient's PHR data. Finally, the patient or doctor updates the PHR data. The specific operations in the above three different workflows are mainly introduced in detail: the encryption and uploading operations performed in the storing PHR data process; the downloading, re-encryption, and decryption operations performed in the requesting PHR data process; the authorization setting and the verification of only being added without change operations performed in the process of updating PHR data process. The main notations used in this section are shown in Table 2.

3.2.1. Storing PHR Data. The interaction process of storing PHR data is shown in Figure 2. The main execution process is as follows:

(1) KeyGen1 (PID) $\longrightarrow$ (pk_{pid}, sk_{pid}): the KeyGen1 algorithm is used to delegate a trusted party to generate a pair of keys. It takes as input the patient's identity PID and outputs a key pair pk_{pid}, sk_{pid}.

(2) KeyGen2 (Num) $\longrightarrow$ K: the KeyGen2 algorithm is used to delegate a trusted party to generate a symmetric key to encrypt the PHR file. It takes as input the specified length of the key $\text{Num} \in \{128, 192, 256\}$ and outputs symmetric key K.

(3) Enc1 (K, C) $\longrightarrow$ C_1: the Enc1 algorithm uses K to encrypt the patient's uploaded file based on the symmetric encryption algorithm AES. It takes as input the symmetric key K and the original PHR file C and outputs the encrypted file C_1.

(4) Enc2 (pk_{pid}, K) $\longrightarrow$ C_2: the Enc2 algorithm uses pk_{pid} to encrypt the symmetric key and writes the encrypted key to the file. It takes as input the patient's public key pk_{pid} and symmetric key K and outputs the encrypted file C_2.

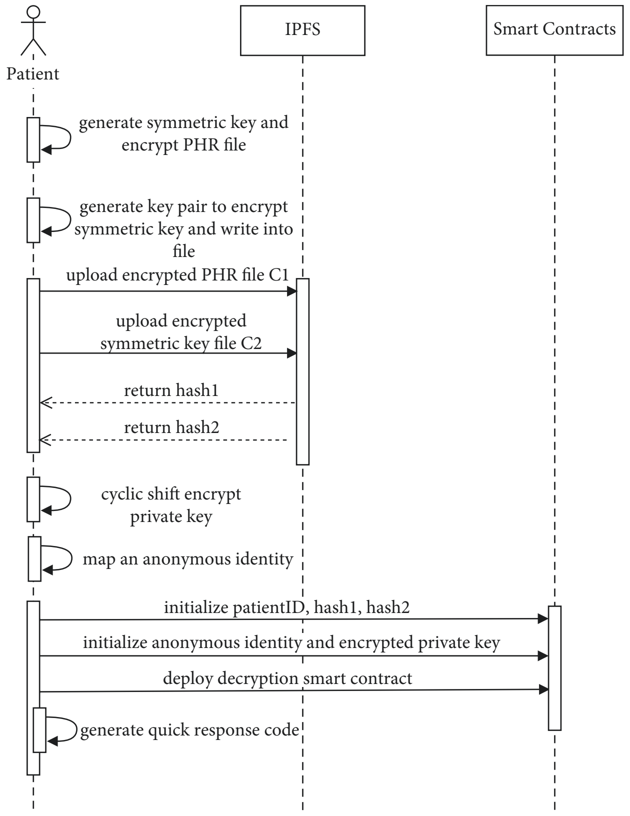
Patient
IPFS
Smart Contracts
generate symmetric key and encrypt PHR file
generate key pair to encrypt symmetric key and write into file
upload encrypted PHR file C1
upload encrypted symmetric key file C2
return hash1
return hash2
cyclic shift encrypt private key
map an anonymous identity
initialize patientID, hash1, hash2
initialize anonymous identity and encrypted private key
deploy decryption smart contract
generate quick response code

Figure 2: Sequence diagram of storing a PHR.

(5) Enc3 (sk_{pid}) $\longrightarrow$ Ensk_{pid}: the Enc3 algorithm is run by those who need to share their PHR data based on the cyclic shift encryption algorithm. It takes as input the patient's private key sk_{pid} and outputs the encrypted key Ensk_{pid}.

(6) UploadToIPFS (C_1, C_2) $\longrightarrow$ (hash_1, hash_2): the UploadToIPFS algorithm is run by patients to upload encrypted files to the distributed storage system IPFS. It takes as input encrypted file C_1 and C_2 and outputs two hash values hash $_1$ and hash_2 which are returned by IPFS.

(7) IdentityMap (PID) $\longrightarrow$ $\text{AnonymityID}_{\text{pid}}$: the IdentityMap algorithm anonymizes the patient's identity, which is only visible to developers after encapsulation. It takes as input the patient's identity PID and outputs an anonymous identity $\text{AnonymityID}_{\text{pid}}$.

(8) InitializeP (PID, hash_1, hash_2, $\text{AnonymityID}_{\text{pid}}$, Ensk_{pid}): the InitializeP algorithm is run by smart contract and stores this information on the blockchain in the form of key-value pairs like {PID, {hash_1, hash_2}} and {$\text{AnonymityID}_{\text{pid}}$, Ensk_{pid}}. The encrypted private key is stored anonymously to prevent the private key from being leaked.

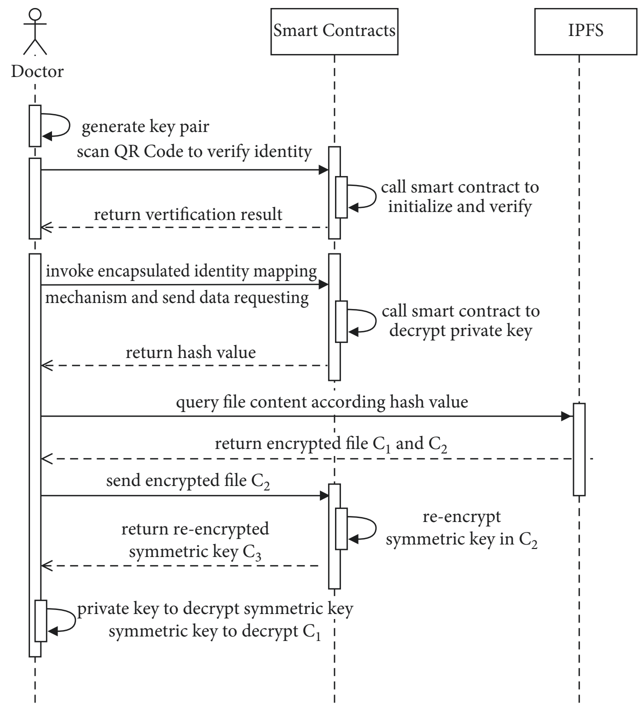
Doctor
Smart Contracts
IPFS
generate key pair
scan QR Code to verify identity
call smart contract to initialize and verify
return vertification result
invoke encapsulated identity mapping mechanism and send data requesting
call smart contract to decrypt private key
return hash value
query file content according hash value
return encrypted file C1 and C2
send encrypted file C2
re-encrypt symmetric key in C2
return re-encrypted symmetric key C3
private key to decrypt symmetric key
symmetric key to decrypt C1

Figure 3: Sequence diagram of requesting a PHR.

After uploading the PHR data, the patient needs to deploy a smart contract for decrypting Ensk_{pid} on the blockchain. The smart contract can be invoked to obtain the decrypted private key. Since the encrypted private key is stored anonymously, the attacker needs a lot of computing power to map the private key to the corresponding user. Meanwhile, the blockchain records who invokes the decryption smart contract. If there is an illegal operation, the patient can pursue the legal responsibility of the attacker after regaining consciousness. At last, the patient can generate a QR code representing his/her identity according to the input PID and the format of the uploaded file, which is convenient for doctors to enter the system quickly.

3.2.2. Requesting PHR Data. The interaction process of requesting PHR data is shown in Figure 3. The doctor enters the system to authenticate his/her identity by scanning the QR code. The QR code can provide the patient's basic information like PID. The main execution process is as follows:

(1) KeyGen3 (DID) $\longrightarrow$ (pk_{pid}, sk_{pid}): the KeyGen3 algorithm is used to delegate a trusted party to generate a pair of keys. It takes as input the doctor's identity DID and outputs a key pair pk_{pid}, sk_{pid}.

(2) InitializeD (DID, HNumber): the InitializeD algorithm is run by a smart contract and stores doctor's information on the blockchain in the form of key-value pairs like {DID, Hnumber}.

(3) Authentication (DID) $\longrightarrow$ True/False: the authentication algorithm is run by a smart contract and used to authenticate the doctor's identity. Firstly, the algorithm judges whether the doctor's ID contains "ED," "ed," "Ed," or "eD" and then checks whether the qualified doctor's ID is in the access control list. This algorithm takes as input the doctor's identity DID and outputs the authentication result true or false.

(4) QueryInfo (PID) $\longrightarrow$ (hash$_1$, hash$_2$): the QueryInfo algorithm is to query the corresponding information on the blockchain. It takes as input the patient's identity PID and outputs hash values of encrypted files hash$_1$ and hash$_2$.

(5) DownloadFile (hash$_1$, hash$_2$) $\longrightarrow$ (C_1, C_2): the DownloadFile algorithm is to download the patient's encrypted files from IPFS by hash values. It takes as input files' hash values hash$_1$ and hash$_2$ and outputs encrypted files C_1 and C_2.

(6) Dec1 (PID) $\longrightarrow$ sk_{pid}: the Dec1 algorithm maps the patient's anonymous identity according to the encapsulated identity mapping algorithm IdentityMap (PID) and then queries the patient's encrypted private key Ensk_{pid} on the blockchain according to this anonymous identity. The algorithm is run by a smart contract to automatically decrypt the patient's encrypted private key. This algorithm takes as input the patient's identity PID and outputs the patient's private key sk_{pid}.

(7) ReEnc (C_2, pk_{pid}, sk_{pid}) $\longrightarrow$ C_3: the ReEnc algorithm uses pk_{pid} and sk_{pid} to generate new key to reencrypt file C_2. This algorithm is run by a smart contract. It takes as input file C $_2$, doctor's public key pk_{pid}, and patient's private key sk_{pid} and outputs reencrypted file C_3.

(8) Dec2 (C_1, C_3, sk_{pid}) $\longrightarrow$ C: the Dec2 algorithm is used to decrypt the file to obtain the complete PHR file. This algorithm uses sk_{pid} to decrypt file C_3 to obtain K and then uses K to decrypt file C_1 to obtain the original PHR file C.

3.2.3. Updating PHR Data. The interaction process of updating PHR data is shown in Figure 4. Patients can update the PHR data themselves or they can authorize physicians to update through settings. The following main execution process is introduced using patient updating as an example.

(1) AuthorizeSet (PID) $\longrightarrow$ True/False: the AuthorizeSet algorithm is for the conscious patient to set the doctor's permission of updating his/her PHR data. This algorithm can reset the authorization setting when necessary for the patient. It takes as input the patient's identity PID and outputs authorization setting result as true or false.

(2) Enc1 (K, C') $\longrightarrow$ C_1': the Enc1 algorithm is the same as the algorithm in the third step of the process of storing PHR data; only the encrypted data is different. It takes as input the symmetric key K and updated PHR file C' and outputs the encrypted file C_1'.

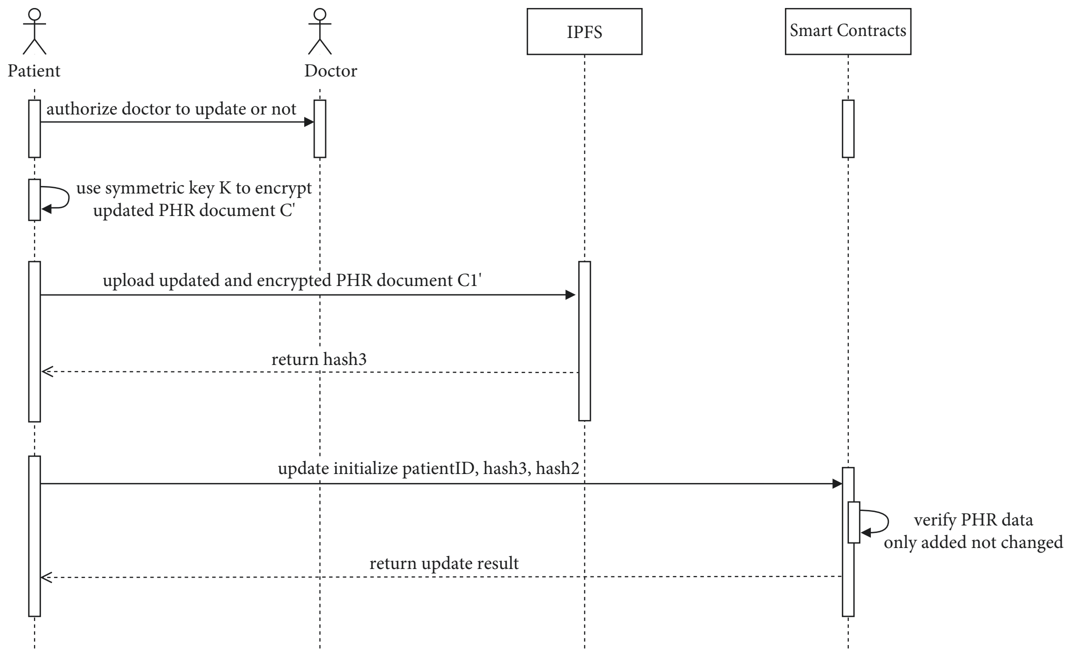
Patient
Doctor
IPFS
Smart Contracts
authorize doctor to update or not
use symmetric key K to encrypt updated PHR document C'
upload updated and encrypted PHR document C1'
return hash3
update initialize patientID, hash3, hash2
verify PHR data only added not changed
return update result

Figure 4: Sequence diagram of updating a PHR.

(3) UploadToIPFS (C_1') $\longrightarrow$ hash$_3$: the UploadToIPFS algorithm is the same as the algorithm in the fourth step of the process of storing PHR data; only the uploaded file content is different. It takes as input the encrypted file C_1' and outputs the hash value hash$_3$ returned by IPFS.

(4) VerifyOnlyAdded (C_1', C_1) $\longrightarrow$ True/False: the VerifyOnlyAdded algorithm is run by a smart contract and verifies the updated PHR file before updating data on the blockchain. This algorithm is to compare the contents of two encrypted files to determine whether the PHR file has only been added but not changed. It takes as input encrypted files C_1' and C_1 and outputs the validation results as true or false.

(5) InitializeP (PID, hash$_3$, hash$_2$, AnonymityID$_{\text{pid}}$, Ensk$_{\text{pid}}$): the InitializeP algorithm can only run if the returned result of the VerifyOnlyAdded (C_1', C_1) algorithm is true. This initialization algorithm just updates the hash value of the saved PHR file, and the rest of the parameters remain unchanged. The data stored in key-value pairs on the blockchain is modified to {PID, {hash$_3$, hash$_2$}} and {AnonymityID$_{\text{pid}}$, Ensk_{pid}}.

For doctors, they enter the PHR data acquisition system after scanning the patient's QR code. If the current patient has authorized the doctor to update, the doctor can click the updating entry to update the data according to the above process. Otherwise, there is no update entry in the PHR data acquisition system.

3.3. System Optimization. In order to ensure that the response speed of the system meets the need in emergency situations, our system has made the following two optimizations.

Figure 5: An example of the QR code.

3.3.1. Quick Response Code. The patient generates a QR code according to his/her ID and the format of the uploaded PHR file. The QR code must include the PHR file format because only the content of the file can be queried from IPFS based on the corresponding hash value. If the content is not saved in the correct formatted file, the content will be garbled. Figure 5 is an example of the QR code. QR code does not store private information, so there is no issue of privacy leakage. The patient can print out his/her QR code and carry it with him/her. In case of an emergency, legal doctors can directly scan the code to enter the doctor authentication interface, saving time for the doctor to find and input the patient's identity information. Furthermore, this QR code can provide effective identity information for the patient when the patient does not bring the identity card or medical card, which brings convenience to the medical treatment.

3.3.2. Bloom Filter. When a doctor applies for access to the patient's PHR, he first needs to authenticate the identity. 100,000 emergency doctor IDs are initialized in the access

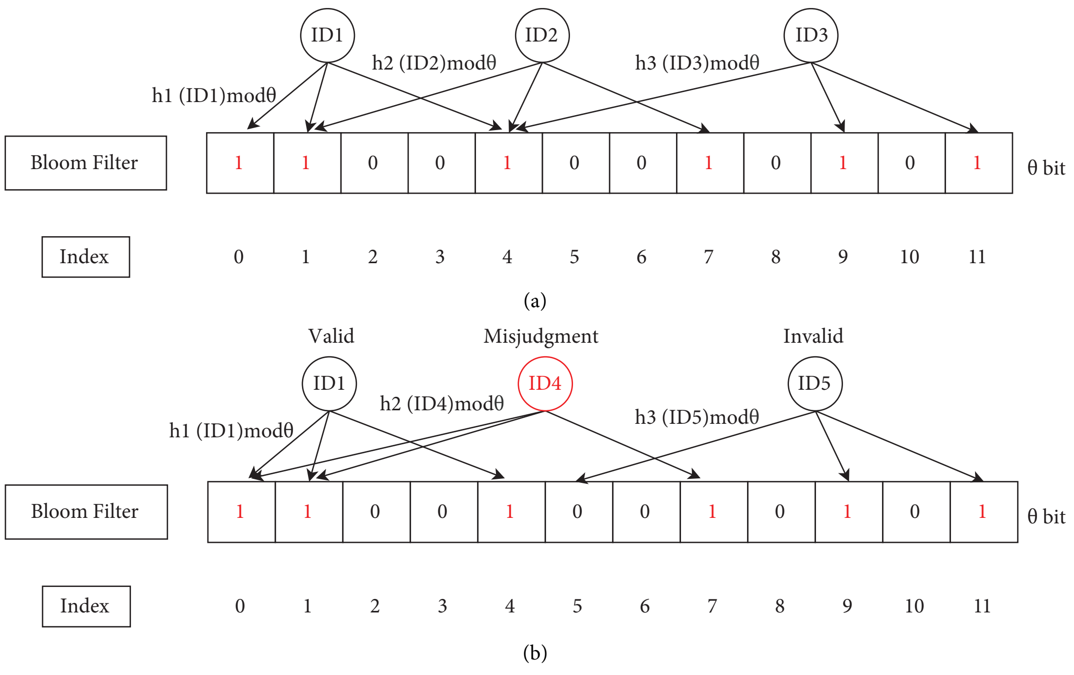

Figure 6: The bloom filter authentication.

control list. If the doctor's input ID is in the access control list, it means that this doctor is authorized. There are too many IDs in the access control list. If the common traversal algorithm is used to query whether the input ID is in the access control list, it is very time-consuming, especially, when multiple doctors use the system to search at the same time. We choose to use the bloom filter to store the doctor's ID, which can effectively shorten the authentication time and speed up the system response. Actually, the bloom filter is a binary array. To judge whether the data is stored in this array is to see if the corresponding bit is 1 or 0 after hash processing; if the corresponding bit is 1, it exists, and if the corresponding bit is 0, it does not exist. The bloom filter authentication process is shown in Figure 6.

(i) Figure 6(a) shows the process of a new ID to be inserted into the bloom filter, which is also the initialization of the access control list. The bloom filter has θ bits. Taking ID_1, ID_2, and ID $_3$ as an example, assume that $k = 3$ hash functions need to be calculated. Each ID is hashed and modelled to get the corresponding value, and the value of the corresponding index in the bloom filter is then set as 1. Taking $k = 3$ as an example, h_1 (ID_1) mod $\theta = 0$, h_2 (ID_1) mod $\theta = 1$, h_3 (ID_1) mod $\theta = 4$, then the indexes corresponding to ID_1 are set as 0, 1, and 4. In the same way, the indexes corresponding to ID_2 and ID_3 can also be calculated and initialized to 1, respectively.

(ii) Figure 6(b) presents the data query verification process, which determines whether a doctor's ID is legal to access the requesting data. In order to determine whether an ID exists in the access control list, the bloom filter calculates the k hash value of the requesting doctor's ID based on the different hash functions. In this example, the filter calculates the three hash values and checks whether the corresponding bits are all 1. When all the verified bits are 1, then the ID is authorized to access the requesting data. Suppose that there are two doctors ID $_1$ and ID_5 who request the same patient's record; we verify their query requests one by one. We first calculate three hash values of ID_1 and take their modulus and then find that the three corresponding index values in the bloom filter are 1, showing that ID_1 is a legal doctor to access the data. Similarly, we calculate the three hash indexes for ID_5 and the corresponding hash values are 0, 1, and 1, resulting in the query denial from ID_5.

Following the designed authentication method based on the bloom filter, the time complexity to set and verify the filter for an ID is $O(k)$, regardless of the amount of data. Here, the parameter k is the number of hash functions, which is generally not too large, and we set $k = 5$ in the system implemented in this paper. Comparatively, the time complexity of traditional traversal methods is $O(n)$, where n is the total number of legal IDs in the access control list. Therefore, the proposed design of the bloom filter can accelerate the authentication speed, especially when the access list contains a large number of legal IDs.

Meanwhile, the drawback of the proposed bloom filter is that there exists a certain false positive rate. For example, given an illegal doctor identity ID_4, the calculated indexes are 0, 1, and 7 which are set as 1 for ID_1 and ID_2. In this case, ID_4 will be treated as a legal doctor, resulting in a false positive case happening. The false positive rate decreases with the number of hash functions, and when we set $k = 5$, the false positive rate is approximately 0.01, which is acceptable in our application scene.

4. Security and Privacy Analysis

In this section, we analyze how the proposed system ensures data security and privacy issues against several possible attacks.

Case 1. The proposed PHR system can prevent attackers from illegally obtaining private keys to decrypt data.

Threat model: the PHR system escrows the patient's encrypted private key on the blockchain. The smart contract is invoked to decrypt the patient's private key. Since the smart contract is transparent, the attacker can obtain the private key of the corresponding patient by downloading the smart contract on the blockchain and performing the decryption process locally.

Argument: the proposed system designs an anonymous identity mapping mechanism, which is only visible to the patient himself/herself. The patient stores his/her encrypted private key on the blockchain using an anonymous identity. Even if the attacker downloads the decryption smart contract deployed on the blockchain to obtain the private key, the attacker does not know which patient's data can be decrypted by the obtained private key. Unless the attacker performs countless tests, this operation will consume a lot of computing power. Therefore, the anonymous mapping mechanism can well protect the privacy of patients.

Case 2. The proposed PHR system can resist malicious access.

Threat model: attackers use the PHR system access mechanism that does not require authorization in an emergency situation to obtain the patient's personal health records by maliciously accessing the system, resulting in leakage of patient privacy.

Argument: the proposed PHR system is based on the Hyperledger Fabric, which is a permissioned blockchain. Different from the public blockchain, Fabric does not allow all users to access it and only targets a limited number of third-party organizations to access it. Moreover, our system also creates an ACL. If someone wants to access PHR data, they need to authenticate their identity after entering the system. Besides restricting access, it is necessary to record the access logs of people entering the system. If the attacker breaks through the front barrier and enters the system, all his/her access and operation behaviors will be recorded on the blockchain. Patients can view access logs of their PHR data when they are conscious. If patients find some suspicious behaviors, they can assert their rights through the law.

Case 3. The proposed PHR system can resist the attacker to tamper with data.

Threat model: personal health records include the patients' physical indicators, as well as their previous diagnosis records, medication status, and treatment plan. The attacker's goal is to modify or replace part of the data, thereby causing damage to the PHR data.

Argument: the PHR data that uploads to IPFS is encrypted by symmetric encryption and asymmetric encryption algorithms. The hash value returned by IPFS is stored on the blockchain. Since the blockchain is open and transparent, the attacker can obtain the hash value from the blockchain and query the corresponding file in IPFS, but the file is encrypted. The encrypted file can only be decrypted with the patient's private key. If the attacker wants to modify the content of an encrypted file, he/she needs to reupload the modified file to IPFS to obtain a new hash value and then modify the hash value on the blockchain. Modifying the data on the blockchain needs to create another main chain, which is almost impossible. In addition, for patients and physicians who have permission to update the PHR data, a blockchain-based verification smart contract is deployed that only allows additions and does not allow modification of previously existing content. Therefore, the model effectively resists malicious tampering with the patient's PHR data.

Case 4. The proposed PHR system can resist single-node attacks.

Threat model: the attacker unites multiple computers to launch an attack on a target so that the power of the attack increases exponentially. Or the attacker node is disguised as multiple identities, and the data that needed to be backed up to multiple nodes is deceptively backed up to the same malicious node.

Argument: since the PHR system is a distributed system based on the blockchain, the patient's PHR data is actually stored in IPFS. IPFS is a decentralized storage system that breaks files into countless fragments and stores them in each node. As more and more nodes are added, single-node attacks will have no impact on the system. In addition, in the model proposed in this paper, a single attacker node cannot disguise multiple identities. Firstly, Hyperledger Fabric needs to verify the identity of each node. Secondly, the newly added node in IPFS needs to complete the replication proof and the space-time proof to prove that it is an effective and stable node; otherwise, it will be challenged and punished by the system. Therefore, the model effectively resists single-node attacks.

Through the analysis of the above four cases, we can know that the system proposed in this paper has high security and privacy. It can resist multiple security attacks and ensure the safety of patients' PHR information. The PHR system stores personal private data, including a lot of sensitive information. Therefore, the security and privacy issues of the PHR system have always been a point of concern for users, which is also the reason why the PHR system has not been widely used. The development of this system will further increase the utilization rate of the PHR system.

5. Performance Evaluation

This section mainly introduces the experimental test environment and evaluates the performance of our system according to the results of the experiments. In addition, we compare our experimental results with the systems proposed in other papers, which shows the superiority of the system proposed in this paper.

The experimental test environment is as follows. The host is a machine with Intel(R) Core(TM) i7-6700 CPU, 2.60 GHz, and 8 GB RAM, and the operating system is Ubuntu 16.04LTS, 64 bits. Front-end interaction is implemented by IntelliJ IDEA, Java 1.8, Java security library, Spring boot 2.4.4 [26], and Apache Tomcat 9.0.44. The simulation experiment test tool used is JMeter. The PHR data set used in the experiment comes from [27], and some medical videos come from [28]. For the testing of the blockchain service, the Hyperledger Fabric network is created in Docker [29] environment with Java JDK. The Fabric network includes an endorser node, an orderer node, and two peer nodes.

Experiment 1. Response time for each stage of a single user.

In order to better present the results of the experimental tests and the superiority of the system, we divide the experimental process into three stages, namely, patient uploading PHR data, patient updating PHR data (here, we take patient updating as an example, and the doctor updating is the same), doctor identity authentication, and doctor getting PHR data. Firstly, we test the time spent in three different transaction stages in the case of a single user. The PHR file format uploaded by the patient is pdf and the file size is 2 MB. The format of the updated PHR file is also pdf and the updated file size is 2.4 MB. The experimental results are shown in Table 3. It can be seen that, under the abovementioned conditions, the process of requesting data by the doctor including authentication and getting the patient's PHR data probably only takes about 7.5 seconds.

We compare our experimental results with the experimental results of the system proposed in [13]. The system proposed in [13] is also a PHR access control system based on Hyperledger Fabric in consideration of the emergency situation. This system does not achieve complete privacy protection and response speed optimization. In addition, the access control strategy of this system is role-based access, which means that the emergency doctor has the right to directly access the data at any time. Comparison finds that, in our proposed system, the speed of emergency doctor authentication and getting patient data is greatly improved. The time taken by emergency physicians from entering the system to obtain the complete patient's personal health records has been reduced by approximately 45%. Moreover, the average response time of receiving messages from the emergency contacts in the traditional access control PHR systems [22, 23] is "431.28 s" during simultaneous conversations. It can be seen that, compared to the blockchain-based system proposed in the other literatures and the traditional system, the performance of the system proposed in this paper is superior.

Table 3: The response time of different transaction.

Roles	Transaction	Response time (ms)	Total time (ms)
The patient	Patient upload PHR	3750	3750
	Update patient's PHR	4183	4183
The doctor	Doctor authentication	2498	7547
	Get patient's PHR	5049	

Experiment 2. Comparison of the execution time of each stage when the data size is different under a single user.

For purpose of more intuitively showing the influencing factors that affect the change of the response time of the system at each stage, we conducted the following tests on the system. Taking a single user as an example, the experiment is designed to test the change of response time with the size of the data block in the four transaction processes of patient uploading PHR data, patient update date, doctor identity authentication, and doctor getting data. The data sizes of 64 kB, 128 kB, 512 kB, 2 MB, 8 MB, 32 MB, and 128 MB are tested, respectively, to get the response time at different transaction stages. For the update stage of patient data, the size of the updated data is slightly larger than the corresponding original uploaded data. The size of the updated data only affects the time to update the data on the blockchain and does not affect the time to verify whether the data is modified, because the part verified for the updated data is always the same part as the original uploaded data. Moreover, the update part of the experiment is based on the successful update as an example.

The experimental results are shown in Figure 7. Figure 7(a) shows that the time for patients to upload and update PHR data rises with the increase of the PHR data size. When the data size is the same, the time of updating the PHR data stage almost and always exceeds the time of uploading stage. The time gap between the two stages increases with the size of PHR data, because the difference is the time that it takes for the smart contract to verify whether the updated data follows the principle of only being added but not changed, and the time of uploading data to the blockchain and updating the data in the block of the blockchain is almost the same. The verification part uses the file comparison algorithm, so the verification time rises with the increase of the PHR data size. In Figure 7(b), we can observe that the response time for doctor identity authentication remains stable basically and does not change with the size of the PHR data uploaded by the patient. This shows that the response time of the doctor authentication stage is independent of the size of the PHR data uploaded by the patient. Figure 7(c) shows that the response time of the doctor getting the PHR data stage rises with the increase of the data size. Meanwhile, the response time increases greatly with the size of the data changing greatly. When comparing the line graph of the patient uploading the PHR data stage in Figure 7(a) and the line graph in Figure 7(c), the following conclusions can be drawn. When the size of data is small, the doctor getting PHR data stage takes longer than the uploading stage, because the doctor getting PHR data stage includes multiple processes of decrypting the patient's

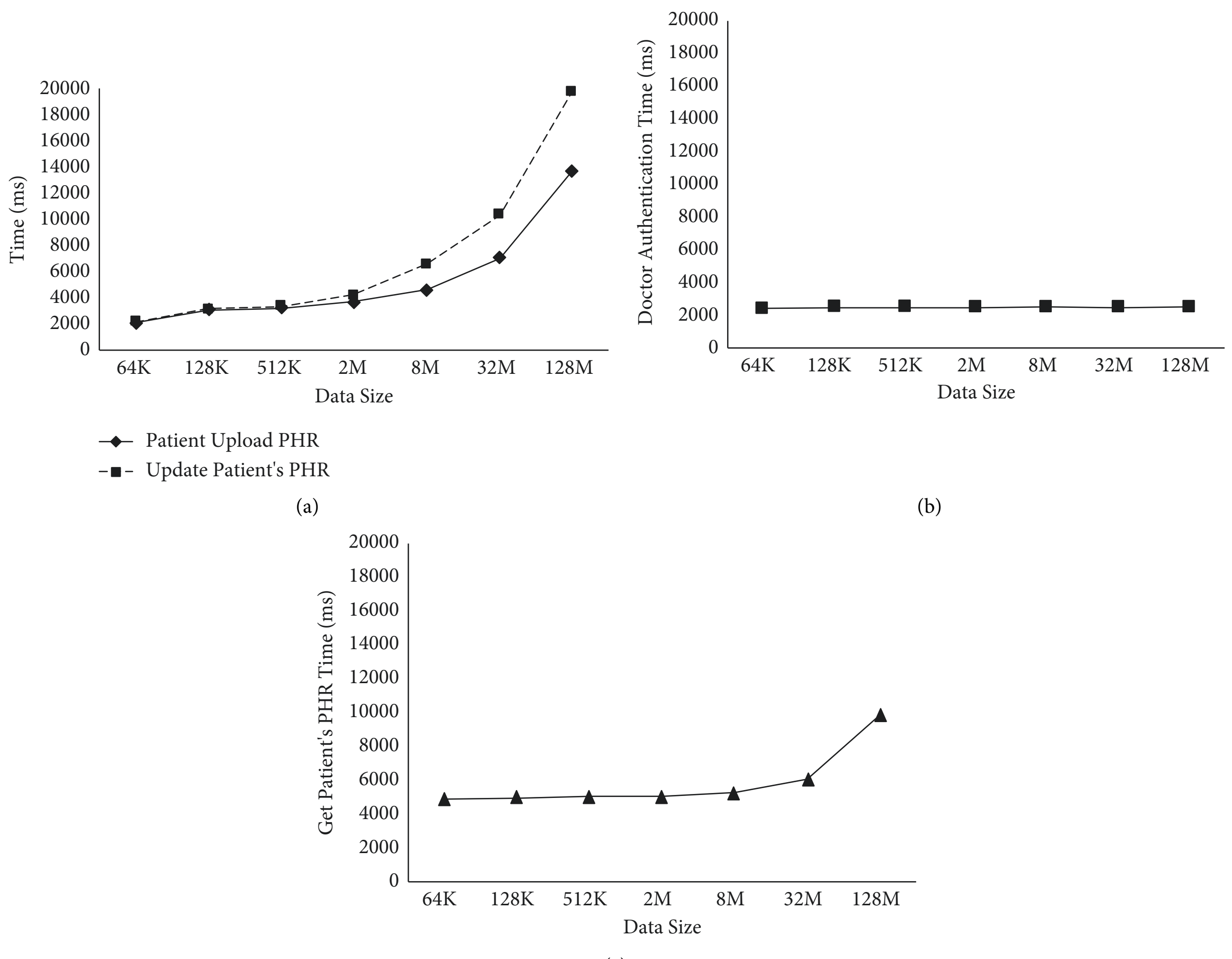

Figure 7: The response time of different data size.

private key, reencrypting the data by the smart contract, and decrypting the data by doctor's private key. When the size of data is large, the patient uploading data stage takes longer than the doctor getting PHR data stage, because the data needs to be encrypted in the uploading stage. The larger the data block, the longer the encryption time. Therefore, it can be known through experiments that the data size is one of the important factors affecting the response speed of the system.

Experiment 3. Comparison of doctor authentication time with and without bloom filter.

To verify the optimization of the response speed of the system designed in this paper, we adopted the following experimental scheme. We compare the time to authenticate the doctor's identity using the bloom filter and using the traditional traversal method without using the bloom filter in the case of different numbers of users. We simulate and test the response time of doctor identity authentication when the number of concurrent users is 20, 40, 60, 80, 100, 120, 140, and 160. Since the emergency system does not have very large-scale concurrency, the maximum number of concurrent users that we choose is only 160.

The experimental results are shown in Figure 8. We can see that as the number of users increases, the advantages of using the bloom filter become more and more obvious. According to the experimental results, when the number of users is 20, the authentication time of the system using the bloom filter is shorter than that of the unused system. As the number of concurrent users gradually increases, the difference between the identity authentication time of doctors using the bloom filter system and the unused system is getting bigger and bigger. Comparing the experimental results horizontally, we can find that the advantages of using the bloom filter gradually become apparent as the number of concurrent users increases.

It can be seen from three experiments that the performance of the system proposed in this paper is better than some systems proposed in the other literatures, so the optimization method that we proposed is effective.

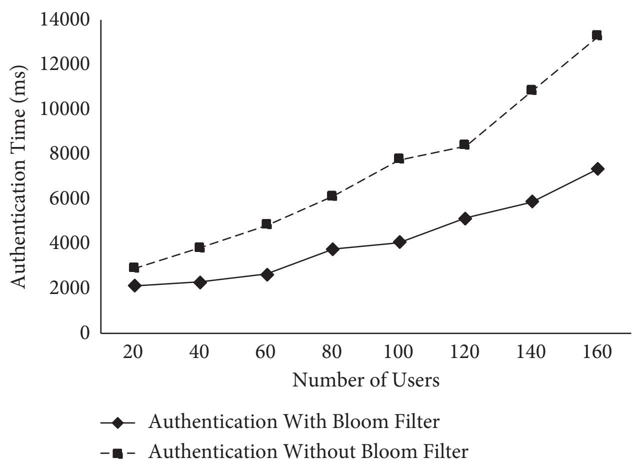

FIGURE 8: The authentication time comparison.

6. Conclusion

In this work, a personal health record system based on blockchain is proposed to protect the privacy of patients and improve the access control efficiency of their records in emergency situations. The proposed solution uses blockchain and IPFS to store health record data, where IPFS stores encrypted complete data and blockchain stores the data hash to mitigate the data storage cost issue. We use cryptography technologies such as symmetric encryption, asymmetric encryption, and reencryption to ensure the privacy of patient PHR data. We propose private key escrow and access control list to enable emergency doctors to access in emergency situations. In addition, we optimize the response speed of the system by designing a QR code and bloom filter so that emergency doctors can efficiently access patients' records even when they are conscious. The proposed system is evaluated by comparing it with the existing ones in the literature, and the experimental results show that our system can decrease the authentication time by 45%, demonstrating the efficiency of the proposed system.

In future work, we plan to apply the proposed system in a realistic application scenario and collect the runtime performance data to do further evaluations.

Data Availability

Data used in this study is available at https://catalog.data.gov/dataset/va-personal-health-record-sample-data.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by National Key Research and Development Plan (Grant no. 2018YFB1800701), National Natural Science Foundation of China, under Grant nos. 62172085, 61972105, U20B2046, and 62032013, Key-Area Research and Development Program of Guangdong Province (no. 2020B0101090005), and China-Singapore International Joint Research Institute, Guangzhou, China.

References

[1] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands, "Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption," *Journal of the American Medical Informatics Association*, vol. 13, no. 2, pp. 121–126, 2006.

[2] D. B. Lafky and T. A. Horan, "Prospective personal health record use among different user groups: results of a multi-wave study," in *Proceedings of the 41st Hawaii International International Conference on Systems Science Waikoloa*, pp. 1–9, IEEE Computer Society, Hawaii, HI, USA, January 2008.

[3] J. Israelson and E. C. Cankaya, "A hybrid web based personal health record system shielded with comprehensive security," in *Proceedings of the 45th Hawaii International International Conference on Systems Science Maui*, pp. 2958–2968, IEEE Computer Society, Maui HI, USA, January 2012.

[4] C. Wang, X. Xu, D. Shi, and W. Lin, "An efficient cloud-based personal health records system using attribute-based encryption and anonymous multi-receiver identity-based encryption," in *Proceedings of the 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing Guangdong*, pp. 74–81, IEEE Computer Society, China, 2014.

[5] Y. Liu, W. Yu, Z. Ai, G. Xu, L. Zhao, and Z. Tian, "A blockchain-empowered federated learning in healthcare-based cyber physical systems," *IEEE Transactions on Network Science and Engineering*, p. 1, 2022.

[6] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. 3561, pp. 1–11, 2014.

[7] J. Chen, C. Zhang, Y. Yan, and Y. Liu, "FileWallet: a file management system based on IPFS and hyperledger fabric," *Computer Modeling in Engineering and Sciences*, vol. 130, no. 2, pp. 949–966, 2022.

[8] L. Zhang, Y. Ye, and Y. Mu, "Multiauthority access control with anonymous authentication for personal health record," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 156–167, 2021.

[9] S. Murphy, "The advanced encryption standard (AES)," *Information Security Technical Report*, vol. 4, no. 4, pp. 12–17, 1999.

[10] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, and A. D. Caro, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference Porto Portugal*, pp. 1–30, ACM, Porto, Portugal, April 2018.

[12] M. N. Huda, S. Yamada, and N. Sonehara, "Privacy-aware access to patient-controlled personal health records in emergency situations," in *Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare*, pp. 1–6, IEEE, London, UK, April 2009.

[13] A. R. Rajput, Q. Li, M. Taleby Ahvanooey, and I. Masood, "EACMS: emergency access control management system for personal health record based on blockchain," *IEEE Access*, vol. 7, pp. 84304–84317, 2019.

[14] B. Liu and J. Xu, "Access control based on proxy Re-encryption technology for personal health record

systems,"vol. 12239, pp. 411–421, in *Proceedings of the Artificial Intelligence and Security - 6th International Conference, ICAIS*, vol. 12239, pp. 411–421, Springer, Hohhot, China, July 2020.

[15] H. X. Son, H. T. Le, N. Q. T. Tang, H. N. D. Huy, N. Duong-Trung, and H. H. Luong, "Toward a blockchain-based technology in dealing with emergencies in patient-centered healthcare systems,"vol. 12605, pp. 44–56, in *Proceedings of the Mobile, Secure, and Programmable Networking - 6th International Conference, MSPN*, vol. 12605, pp. 44–56, Springer, Paris, France, October 2020.

[16] H. M. Hussien, S. M. Yasin, N. I. Udzir, and M. I. H. Ninggal, "Blockchain-based access control scheme for secure shared personal health records over decentralised storage," *Sensors*, vol. 21, no. 7, p. 2462, 2021.

[17] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[18] T. Bhatia, A. K. Verma, and G. Sharma, "Secure sharing of mobile personal healthcare records using certificateless proxy re-encryption in cloud," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 6, pp. e3309–e3321, 2018.

[19] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, no. 99, pp. 102887–102901, 2019.

[20] T. T. Thwin and S. Vasupongayya, "Blockchain-based access control model to preserve privacy for personal health record systems," *Security and Communication Networks*, vol. 2019, no. 5, pp. 1–15, 2019.

[21] F. Aljumah, R. H. M. Leung, M. Pourzandi, and M. Debbabi, "Emergency mobile access to personal health records stored on an untrusted cloud,"vol. 7798, pp. 30–41, in *Proceedings of the Health Information Science - Second International Conference, HIS*, vol. 7798, pp. 30–41, Springer, London, UK, March 2013.

[22] P. Thummavet and S. Vasupongayya, "A novel personal health record system for handling emergency situations," in *Proceedings of the International Computer Science and Engineering Conference, ICSEC Silpakorn Univ*, pp. 266–271, IEEE, Nakhon Pathom, THAILAND, September 2013.

[23] P. Thummavet and S. Vasupongayya, "Privacy-preserving emergency access control for personal health records," *Maejo International Journal of Science and Technology*, vol. 9, no. 1, pp. 108–120, 2015.

[24] M. M. Madine, K. Salah, R. Jayaraman et al., "Fully decentralized multi-party consent management for secure sharing of patient health records," *IEEE Access*, vol. 8, no. 99, pp. 225777–225791, 2020.

[25] V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "A personal health record system for emergency case management,"vol. 127, pp. 83–96, in *Proceedings of the Biomedical Engineering Systems and Technologies - Third International Joint Conference, BIOSTEC*, vol. 127, pp. 83–96, Springer, Valencia, Spain, January 2010.

[26] A. C. Ríos, A. Boyko, A. Nesterov, A. Clement, and C. Walls, *Spring boot*Spring Inc, Tiruppur, TN, India, 2022.

[27] D. V. O. Usdatagov, Ed., *VA Personal Health Record Sample Data - Data.Gov*, Department of Veterans Affairs, Coimbatore, TN, India, 2022.

[28] M. Roger Seheult and P. C. Kyle Allred, Eds., *CME MedCram - Best Medical Lectures and Medical Videos*, MedCram, LLC, California, CA, USA, 2022.

[29] S. Johnston, J. L. de Morlhon, M. Carter et al., Eds., *Docker*, Docker Inc, Palo Alto, CA, USA, 2022.