Multimedia Tools and Applications (2023) 82:16279–16300
https://doi.org/10.1007/s11042-022-14023-3

Blockchain-based medical health record access control scheme with efficient protection mechanism and patient control

Wen-Xin Yuan[1] · Bin Yan[1] · Wen Li[2] · Liu-Yao Hao[3] · Hong-Mei Yang[4]

Received: 13 August 2021 / Revised: 28 December 2021 / Accepted: 23 September 2022 /
Published online: 7 November 2022

Abstract

The patient's medical health record (PMHR) has always provided a large amount of research data to medical institutions and pharmaceutical companies, etc., and has contributed to the development in medical research. However, such PMHR data contains the patient's personal privacy and should be shared under the control of the patients, not the hospital where this data is acquired. In order to protect the privacy of PMHR data while realizing efficient data sharing, this paper proposes a blockchain-based sharing and protection scheme. In this solution, the PMHR data are encrypted and stored in a cloud server, which is equipped with an access control scheme implemented as a smart contract on a blockchain. Different from previous works, in order to ensure efficient access and reduce the workload of patients, the types of users who can apply for access are limited to hospitals and pharmaceutical companies. In order to resist the potential Man-in-the-middle (MITM) attack, we have introduced an improved proxy re-encryption scheme to ensure the secrecy of PMHR data while reducing the computational complexity. The whole system is implemented using Solidity and tested on 10 nodes for function verification. Experimental result shows that the proposed system is more efficient than previous systems. Security under the MITM attack is also ensured by security analysis.

Keywords Medical health records · Blockchain · Proxy re-encryption

✉ Bin Yan
yanbinhit@hotmail.com

[1] College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao, 266590 People's Republic of China

[2] Confidentiality Administration Bureau of Ji-Ning, Ji-Ning, People's Republic of China

[3] China Mobile Communications Research Institute, Beijing, People's Republic of China

[4] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, People's Republic of China

1 Introduction

The patient's medical health record (PMHR) has always been an indispensable part of the medical system. The storage of PMHR has evolved from the original paper-based medical records (PMRs) to the current electronic medical health records (EMRs). Obviously, EMRs are more convenient and effective in storage, sharing, query, research, etc. [15].

There are three types of electronic health systems, including the traditional centralized storage electronic health systems TEHS, the cloud server-based electronic health systems (CEHS), and the blockchain-based electronic health systems (BEHS).

In the TEHS, PMHR is stored in the medical institution where are generated. The advantage of this approach is that, it is convenient for medical institutions to use and protect the PMHR data. However, the disadvantage is that it cannot realize data sharing with other medical institutions. When a patient is transferred to another medical institution, the patient's historical data cannot be transferred synchronously. A key technical barrier is that different medical institutions use different software and hardware for PMHR storage. As a result, inter-operations between institutions are not supported.

CEHS is aimed to enhance inter-operations between medical institutions. Patients can use cloud storage client to upload their PMHR to the cloud server [8, 20], which improves the medical services quality and development of paperless medical office. However, since all medical data is stored in a cloud server-centric structure, a pure CEHS always has the problem of single point of failure and collusion attacks.

The emergence and development of blockchain technology has attracted the attention of researchers and scholars all over the world [11, 22]. At the same time, electronic health systems based on blockchain have also been developed [10, 13, 14]. The two electronic health systems proposed in the past, whether they are TEHS or CEHS, have problems on sharing interaction or privacy and security risks. In the BEHS, the decentralization and non-tamperability of the blockchain are used to solve the problems existing in the TEHS and CEHS.

Recently, Gan et al. proposes an incentive mechanism for the patients to actively share their medical data [12]. This mechanism motivates the patients to play an active role in medical data protection and sharing. In this mechanism, a cloud servers is used as intermediaries to complete the access and storage of the original patient data. When interacting with visitors, there are still potential security risks when the keys are transmitted through the cloud server. In addition, users must obtain the patient's permission through the access control contract to use the data. There is no restriction on the type of access users, which increases the workload of the patient, and there is also the possibility of malicious access and invalid access.

The advantage of this scheme is to propose an improved access control scheme based on blockchain. Compared with the previously proposed algorithms, this solution uses improved proxy re-encryption to improve the security of PMHR and reduces the time required for the entire encryption and decryption process in order to solve the potential security problems in the patient incentive mechanism. At the same time, the access control scheme proposed by this scheme limits the types of access users, reduces the workload of patients in the process of controlling access, and makes the entire access scheme more reasonable and efficient.

The main contributions of this paper can be summarized as follows:

1. An improved proxy re-encryption scheme is designed to complete the encryption and transmission of PMHR data, which solves the potential key leakage problem in the cloud servers. The computational complexity of data processing is also significantly reduced.

2. This paper proposes a PMHR access control contract based on blockchain, which is suitable for pharmaceutical companies and other medical institutions. The purpose is to reasonably limit the types of users who access media data. Therefore, this solution can avoid malicious and invalid access, and significantly reduce the workload of patients to process requests.

The rest of this paper is structured as follows. Section 2 introduces related works of the medical electronic health system based on blockchain. In Section 3, we describe the details of the proposed method, including access control contracts. Section 4 studies the case of access control contract through Ethereum, and analyzes the feasibility and security of this scheme. Finally, we conclude this paper in Section 5.

2 Related work

In the field of medical research, the real PMHR are self-evident for medical research. Kumari et al. used machine learning classifiers to group predictions of liver disease [19]. Dabral et al. used convolutional neural networks for cancer detection [6]. Negi et al. proposed a deep neural architecture for face mask detection of the simulated masked face dataset for the Covid-19 pandemic [25]. Negi et al. studied the classification of multi-class images of plant diseases based on deep neural networks [24]. Darbari and others put forward the requirements for the artificial intelligence technology awareness of thoracic surgeons [7]. Ansari et al. proposed a hidden Markov model, which can be used for depression detection based on content ratings [3]. Alok et al. proposed a deep learning-based image classifier for malaria cell detection [1]. Kumar et al. proposed a novel color space feature based on superpixels for salient object detection [18]. The above works reflect the importance of PMHR for medical research.

In this section, we briefly review current works related to electronic medical health records (EMRs), including cloud-based servers and blockchain-based electronic health records. Since the EMRs contains the patient's personal sensitive information, it can be the target of security attack. In such a scenario, the current EMRs system may not be fully secure. This security issue is the main motivation of our work.

2.1 Electronic health record based on cloud server

In traditional medical and health record sharing solutions, cloud server storage technology is often used to solve storage problems in the data sharing process [8, 20]. In the cloud server-based medical health record sharing solution, PMHR is stored in the cloud server. Patients, doctors and visitors can download the medical data they need through the cloud server. The system block diagram is shown in Fig. 1 The cloud server-based medical and health record sharing solution completely trust the cloud server. The cloud server may steal or tamper with some patient data, resulting in the leakage of patient data and causing immeasurable losses.

Liu et al. proposed an efficient and secure access control scheme [21], which allows users authorized by the access control scheme to access the EMRs stored in the cloud server. It supports certain doctors to edit items in PMHR. However, there are some security problems with EMRs based on cloud server storage. If the cloud server provider is maliciously attacked, it is likely to cause the leakage of patient data. When the cloud server that provides storage services is paralyzed or down, the cloud server suddenly stops running, and even the PMHR will be lost.

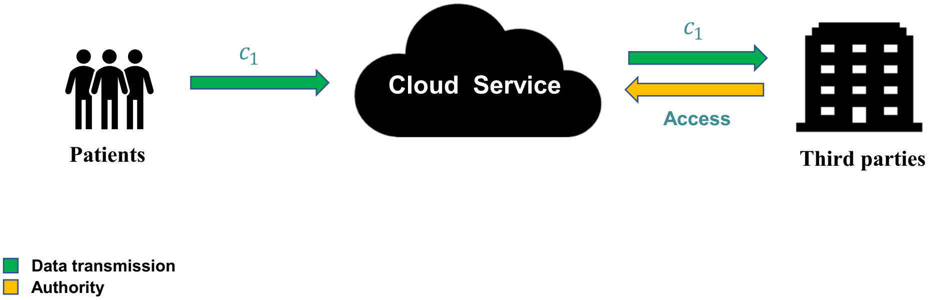


Fig. 1 The block diagram for EMRs system based on cloud server

2.2 Blockchain-based electronic health records

With the development of blockchain technology, its characteristics such as decentralization, immutability and anonymity have been widely explored and utilized. At present, many scholars are paying attention to issues such as sharing of PMHR and privacy protection based on blockchain technology. Karame et al. introduced the technology and application of blockchain in terms of security and privacy [16]. Rabah and others reviewed the opportunities and challenges of blockchain-based EMRs systems. Khezr and others gave a comprehensive outlook on the future development and research of blockchain technology in the medical industry [17].

Chen et al. proposed a storage solution and service framework for medical data storage [4], sharing and use based on blockchain and cloud servers. Tanwar et al. proposed an electronic medical record sharing system based on Hyperledger [30], but its scalability problems are still inevitable. Wang et al. proposed a cloud-assisted electronic health record sharing to achieve security and privacy protection through a consortium chain [32]. Amofa et al. proposed a blockchain architecture to realize the security control of PMHR in the exchange of medical and health information by matching smart contracts with user-generated access strategies [2]. In [23], Mikula et al. proposed an identity and access management system that uses blockchain technology to achieve entity authentication and authorization. This system describes the application of blockchain in the framework of Hyperledger for identity authentication and access management. Although these works have implemented blockchain-based medical data storage and access control systems in different ways, they have not specifically considered the interests of patients.

In order to strengthen the protection of the personal interests of patients, Omar et al. proposed a patient-centric blockchain-based medical data privacy protection platform: the medical chain, which uses the decentralized characteristics of the blockchain to ensure the privacy and integrity of PMHR [27]. Through the blockchain-based medical data management system, the use of blockchain as a storage space realizes the privacy protection of PMHR. Chen et al. proposed a patient-centric blockchain-based model that controls the access of the entire medical institution through smart contracts [5]. It can change the hierarchical structure of healthcare by returning the control authority of the PMHR to the patient. This transfer of authority implements a patient-centric system, but no detailed experimental procedures are given. In [33], a blockchain-based key management protocol for a patient-centric medical information system is proposed. This solution focuses on protecting the key pair that encrypts and decrypts PMHR, so that patients can control PMHR by controlling the use of keys. Gan et al. proposed a blockchain-based access control scheme for electronic health systems [12]. In their solution, the patient plays a supervisory role, allowing the patient's medical institution to directly access the data without prior authorization. Other users and institutions need to apply for access to obtain access rights. In addition, the patient's data is uploaded to the blockchain via the cloud server. The specific system block diagram is shown in Fig. 2.

3 The proposed system

3.1 Motivations

The purpose of this method is to protect the rights and interests of patients by enhancing the safety of PMHR. This method is also used to ensure the normal visits from medical institutions and pharmaceutical companies.

In the previous method, although the patient's control of their own medical and health records was solved, there were still two problems: On the one hand, the cloud server stores PMHR, but in the interaction with medical institutions and third parties, there may be problems such as patient privacy leakage; on the other hand, there are no restrictions on visitors. Any third party can apply for access, which reduces the research and work efficiency of medical institutions and pharmaceutical companies.

This paper proposes to provide an improved proxy re-encryption function for the cloud server. When the encrypted PMHR is uploaded to the cloud server, the ciphertext is sent to the visitor after the proxy re-encryption. This solves the first problem outlined above.

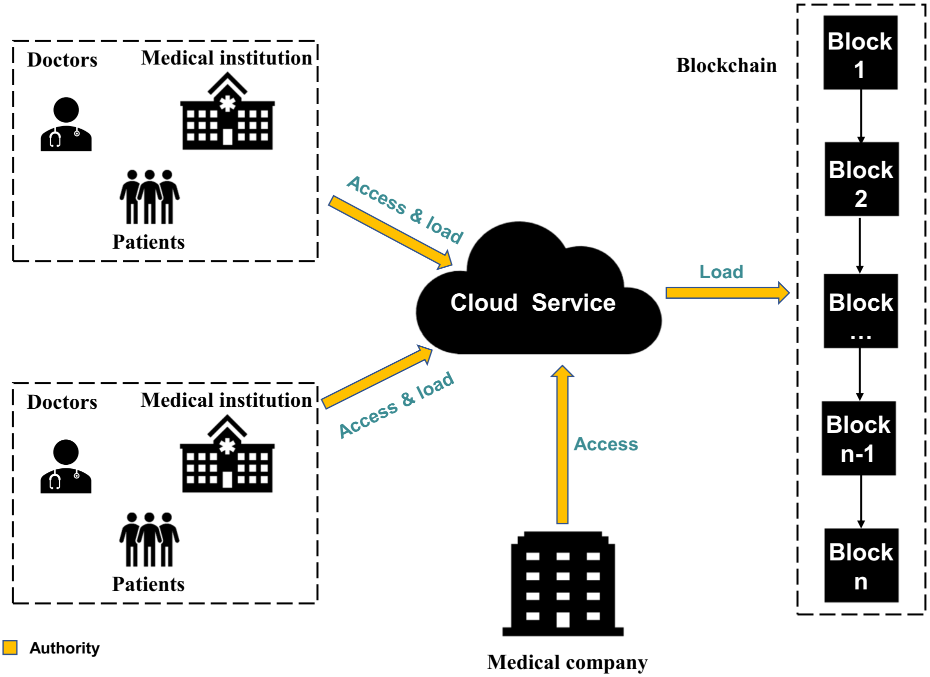


Fig. 2 Blockchain-based electronic health record block diagram

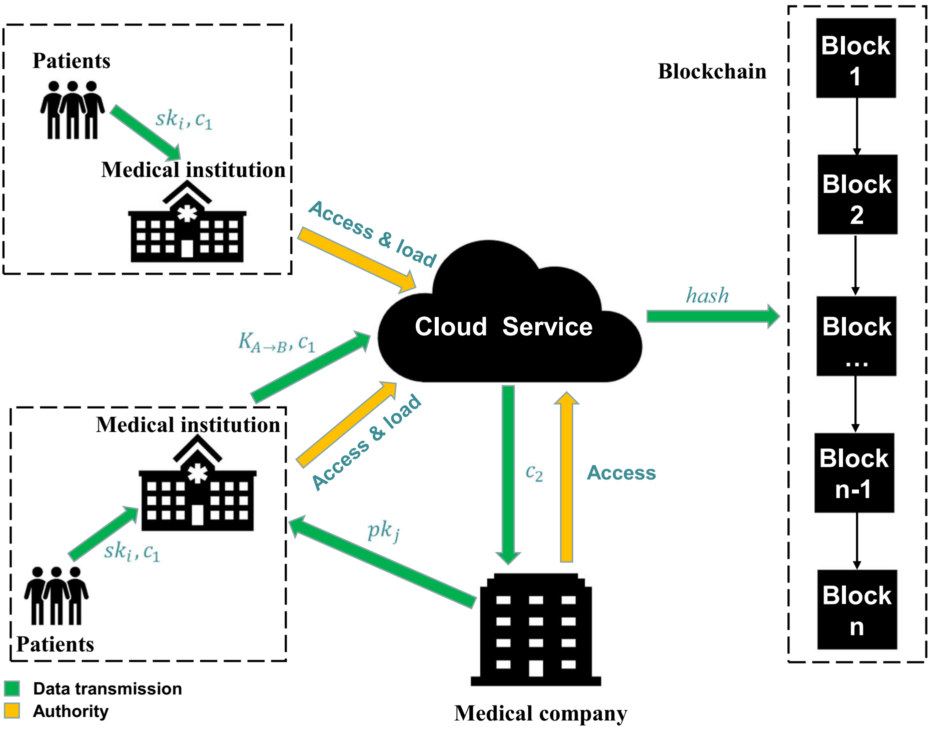


Fig. 3 The schematic diagram of the proposed method

However, the patient's medical institution can still directly access the PMHR without the patient's authorization. The medical institution where patients see the doctors generates the hash value of the PMHR and uploads it to the blockchain for storage. It is changeable and tamper-proof, so patients can check whether their data has been tampered with at any time. When other medical institutions or pharmaceutical companies visit, the applicant for the visitor passes the access control contract. After the patient's authorization is obtained, the cloud server sends the re-encrypted PMHR to the visitor. Patients can supervise the behavior of the requesting visitor throughout the process, and they can stop the visit at any time.

In order to encourage patients to share their PMHR more actively and to improve the efficiency of research and work by medical institutions and pharmaceutical companies, our work adopts an incentive mechanism. The mechanism rewards patients based on the importance of PMHR and the number of times patients share PMHR. This solves the problem that some patients are unwilling to share PMHR and improves the efficiency of the entire system. Figure 3 shows a schematic diagram of the proposed solution.

3.2 User roles

The scheme proposed in this work involves four roles: patients, medical institution, cloud server and other medical institutions or pharmaceutical companies. Figure 4 shows the permission diagram of all roles. The specific work content of each role is as follows.

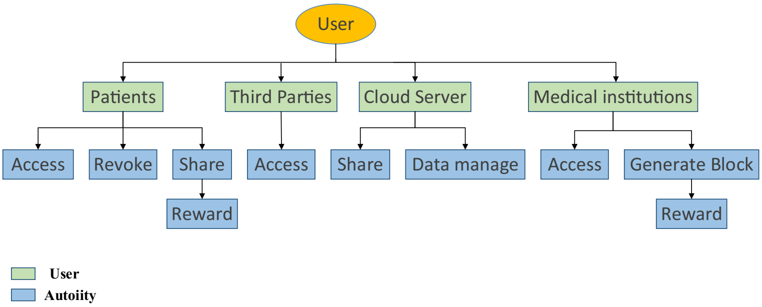


Fig. 4 The flowchart of role authority

1. **Patients** The patients play one of the most important roles in our approach. They are the producers of PMHR and the sole owner of PMHR. They have the right to manage their own PMHR. Patients can encrypt the PMHR by generating their own keys and upload them to the cloud server for storage. Through smart contracts, patients can access their own PMHR at any time. When other institutions want to access these patients' PMHR, the patients can decide whether to share it or not. If the patients agree to share, the institution will obtain the patients' authorization. The incentive contract in the access control contract will reward the patients. This measure is to encourage patients to actively share their PMHR. In addition, if the patients are dissatisfied with the behavior of the institution being visited or for any other reason, the visit can be disabled at any time.
2. **Medical institution** Medical institution plays indispensable roles in this system. It provide diagnosis services for patients, generate the PMHR, and upload the hash value of the encrypted PMHR to the blockchain. The medical institution can generate a session key by requesting the public key of the access institution and the private key of the authorized patient. This session key is sent to the cloud server to perform the proxy re-encryption process. In our scheme, the medical institution can directly view the PMHR without prior authorization. But they cannot tamper with or delete the data. In addition, if the patient interrupts the visit, the medical institution cannot continue to access the PMHR.
3. **Cloud server** The cloud server is used to store the PMHR uploaded by the patient's medical institution, it provides services to achieve the interaction between the patient, the medical institution and the third party. After the patient encrypts and uploads the PMHR, the cloud server stores the encrypted PMHR. When a third party applies for access to the PMHR, the cloud server acts as the data manager and uses proxy re-encryption technology to share PMHR under the premise of ensuring data security. The cloud server only provides computing and session key services. The cloud server cannot obtain the private key of the patient and the visiting organization. Therefore, it cannot view the plaintext of PMHR, this way avoids the risk of leakage of the privacy of the PMHR.
4. **Third parties** Third parties refer to the medical institutions and pharmaceutical companies other than the medical institution where the patient is treated. The third party does not participate in the generation, uploading and block generation of PMHR. Third parties can only apply for access to the PMHR through the access control system. If the patient's permission is obtained, third parties will send his public key to the patient's

medical institution, then they will receive the re-encryption returned by the cloud server through proxy re-encryption. After that, the PMHR can be decrypted with its own private key to obtain the plaintext. Third parties only have the rights to view PMHR to conduct legal work or research on PMHR, but cannot tamper or delete the records. The patients can cancel their access rights of the third parties at any time.

3.3 Data uploading and processing

1. **Improved proxy re-encryption** In order to ensure the safety of data uploading and sharing, this paper proposes an improved proxy re-encryption method. Because the PMHR usually contains data that takes up a lot of space, including B-ultrasound, electrocardiogram, etc. The AES encryption is used to encrypt PMHR, and the RSA encryption is used to encrypt the AES key. Through this method, not only the security of the PMHR is guaranteed, but also the encryption rate is also improved. The patient and the medical institution entrust the cloud server to store encrypted PMHR and encryption key. The cloud server converts the generated encryption key into a ciphertext encrypted with the public key of the third party requesting access to PMHR. Then a third party organization can decrypt the encryption with its own private key to obtain the patient's AES key and the key-encrypted PMHR. Finally the third party can obtain the PMHR through the key decryption. Compared with the traditional proxy re-encryption technology, the encryption method proposed in this scheme improves the efficiency of encrypted data. The characteristic of proxy re-encryption is that in the entire process of requesting access and sending records. PMHR is very safe, and the cloud server cannot view the plaintext of PMHR, this way improves the security of the entire access system. The specific steps are as follows:

    - Key generation : Patient p_i initializes the key pair generator. Patient p_i transmits the parameter RSA through the key pair generator,and randomly generate his own private key sk_i and public key pk_i. The patient p_i transmits the parameter AES through the key generator, and randomly generates its own key k.
    - Encryption : The patient p_i first transmits the medical health record m and the key k through the encrypt method of the AES encryption algorithm to obtain the ciphertext c. Then the patient p_i uses the encrypt method of the RSA encryption algorithm to input the public key pk_i and the key k, the method generates the ciphertext c_1. Finally, the patient p_i sends c, c_1 and pk_i to the medical institution M.
    - Conversion key generation : The medical institution M inputs the public key pk_i of the patient p_i and the private key sk_j of the visitor a_j through the re-key generation method, then the method generates the conversion key $K_{A,B}$.
    - Re-encryption : The cloud server CS uses the re-encrypt method to input the conversion key $K_{A,B}$ and the ciphertext c_1. The output is the ciphertext c_2 encrypted by the visitor a_j. At last, the cloud server CS sends c_2 to the visitor a_j.
    - Decryption : The visitor a_j transmits the ciphertext c_2 and his own private key sk_j through the decrypt method of the RSA decryption algorithm, then a_j decrypts it successfully, obtains the key k. Then the visitor a_j uses the decrypt method of the AES decryption algorithm to input the ciphertext c and the key k. Finally, the visitor a_j gets the plaintext of medical health record m.

2. **Data uploading** Medical institution needs to upload the ciphertext of PMHR to the cloud server, and finally records it on the blockchain in the form of hash value. Since PMHR contains a lot of sensitive information for the patient, the patient will generate a key to encrypt the data before uploading it. In addition, PMHR usually contain data that takes up a lot of space, including B-ultrasound, electrocardiogram, etc. Considering the size of each block in the blockchain, it is inappropriate for these data to be uploaded directly to the blockchain for storage. Therefore, the original encrypted data of these PMHR are stored in the cloud server, and the obtained hash value is uploaded to the blockchain for storage after the hash calculation provided by the medical institution. This not only maximizes the use of the storage function of the blockchain, but also further strengthens the security of PMHR. Figure 5 shows the entire data upload process, the details are as follows.

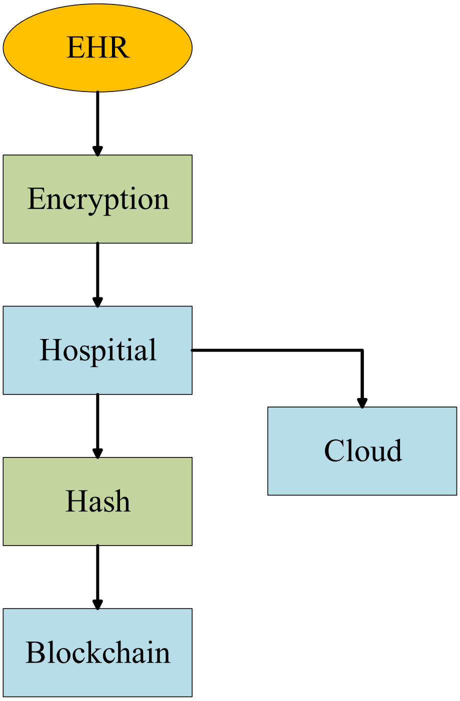
First, the patient generates his own PMHR in the medical institution. Then the patient encrypts the PMHR by generating his own key. After receiving the encrypted PMHR, the medical institution uploads the PMHR to the cloud server. The hash value of the PMHR is calculated through the hash function and is uploaded to the blockchain for storage.

Regarding the encryption processing of PMHR, this article uses a hybrid encryption algorithm of AES and RSA. First, the patients encrypt their medical health record data with AES key, then the patients encrypt the AES key with RSA encryption algorithm.

3.4 Access control scheme

This section introduces the access control scheme. This scheme realizes the interaction between the patient and the visiting organizations. The patient can control PMHR through

Fig. 5 Flowchart for data uploading



the access control scheme. The visiting organizations can apply for access through the registration information of this scheme. The access control scheme proposed here is implemented through smart contracts. Each contract completes its own function. The following will introduce the overall block diagram of the access control scheme and the role of each smart contract.

According to each user and corresponding authority in Fig. 4, the overall block diagram of the access control scheme can be illustrated in Fig. 6. There are four types of contracts in total, including the registration contract, the access contract, the incentive contract, and the revoke contract. The explanation of each symbol in the smart contract is as follows:

1. **userAddress** It represents the value of the registered user address.
2. **dataHashes** It is used to map user address using hash value.
3. **recordList** It indicates the result of the visitor's request for access.
4. **approvedIRs** The result corresponding to the user address that the patient is allowed to access
5. **getRewardN** It is used to calculate the reward corresponding to the number of patients sharing PMHR.
6. **getRewardQ** It is used to calculate the reward corresponding to the quality of PMHR score.

1. **Registration contract** In the access control scheme, the first step that any institution needs to do is to register its own institution's information. Each institution enters its own account address through the registration contract so that the system can obtain the institution's detailed information. This is done for patients and medical institutions can fully understand the specific information of the visiting organization. The Algorithm 1 of user registration information is shown below.

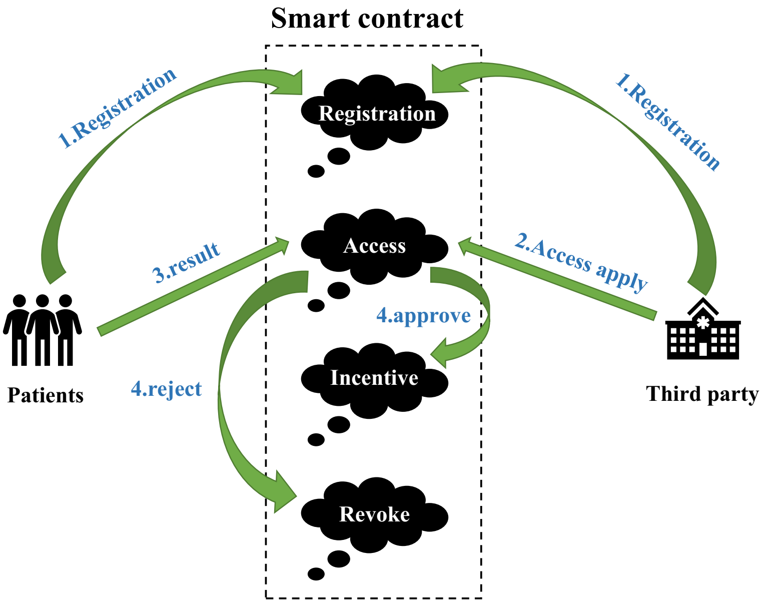


Fig. 6 Block diagram of access control contract

```
Input:
    User personal address $\mathbf{U}$
Output:
    userAddress $\mathbf{D}$
    /* $msg.sender$ is the default address of the contract calling user
    When $\mathbf{D} = 0$ means registration failed */
if $\mathbf{U} = msg.sender$ then
    $\mathbf{D} = \mathbf{U}$
else
    $\mathbf{D} = 0$
end if
```

Algorithm 1 User registration.

2. **Access contract** The access contract is used by the patient to stipulate the authority of each access role. Specifically, there are different jurisdictions for patients' medical institutions, other medical institutions, and pharmaceutical companies. Patients have the highest access rights to their own medical and health records, including relevant images, data, and all patient-related data. The patient's medical institution is the role that generates the patient's medical health record. It can view and modify the medical health record according to the patient's condition. If other institutions or companies obtain the patient's permission, the patient will authorize these approved institutions, and they can view the scanned files of the patient's medical and health records within a limited range. The algorithm of the access contract is shown below.

```
Input:
    userAddress $\mathbf{D}$
Output:
    Access permission $\mathbf{A}$
    /* $patientAddress$ is the value of each patient's address */
if dataHashes[$\mathbf{D}$] == sha256($patientAddress$) then
    recordList[$\mathbf{D}$] = TRUE
    approvedIRs[$\mathbf{D}$] = TRUE
else if approvedIRs[$\mathbf{D}$] = TRUE then
    $\mathbf{A}$ = TRUE
else
    recordList[$\mathbf{D}$] = FALSE
    $\mathbf{A}$ = FALSE
end if
```

Algorithm 2 Access contract.

3. **Incentive contract** Incentive contract is an incentive policy for patients after they agree to share their PMHR. Its main purpose is to encourage patients to actively share their PMHR and provide medical institutions and pharmaceutical companies with more medical research information. Incentive contracts are divided into two situations. One is that the patient responds with authorization after receiving the user's access request, and the other is that the patient actively authorizes access to some institutions. Specifically, the incentive contract uses an incentive function to calculate the reward the patient receives.

This article proposes an incentive mechanism to encourage patients to actively share their PMHR. The proposed method calculates the patient's reward R through the incentive function getIncentive(). We calculate the total reward R that the patient can get based on the following two reward evaluation factors.

(a) **Medical health record quality.** This mainly refers to the research value of the PMHR for medical institutions and pharmaceutical companies, as well as the completeness and correctness of the PMHR. These factors are used to comprehensively determine the quality of PMHR.

(b) **Sharing level.** The sharing level mainly refers to the degree to which patients share their own PMHR, which can be determined by the number of patients actively authorized to visit.

Assume that the total reward of the entire incentive mechanism is R. the reward for the patient through the quality of PMHR is recorded as R_q, and the reward for authorized access by the patient is recorded as R_s. Therefore, the total reward of the entire incentive mechanism is $R = R_q + R_s$. The specific steps of the excitation function algorithm are shown below.

```
Input:
    userAddress $\mathbf{D}$, Number of shared PMHR $\mathbf{N}$, the quality score of PMHR $\mathbf{Q}$
Output:
    Reward $\mathbf{R}$
    /* $patientAddress$ is the value of patient's address*/
if dataHashes[$\mathbf{D}$] == sha256($patientAddress$) then
    $\mathbf{R}_s$ = getRewardN($\mathbf{N}$);
    $\mathbf{R}_q$ = getRewardQ($\mathbf{Q}$);
    $\mathbf{R} = \mathbf{R}_s + \mathbf{R}_q$
else
    $\mathbf{R} = 0$
end if
```

Algorithm 3 Incentive contract.

4. **Revoke contract** The cancellation of the contract is used to determine whether the registered information is valid and whether there is any abnormal behavior. Judging whether the user information is valid and able to continue the following contract mainly depends on the following two abnormal behaviors.

- Incomplete registration information.
- Frequent access requests.

If there is abnormal behavior, the patient can call the following revoke function to deny the access authority of this access institution.

- revokeAccess(). When calling this function, the patient can revoke the requesting visitor's application for access and terminate the institution's access.

Table 1 Implementation blueprints

Environment	Parameters
System	AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz Windows 10 Home 64-bit 16GB
Blockchain	Solidity 0.4.24 Web3.js

4 Results

The method proposed in this article includes the improved proxy re-encryption technology, data upload process, and access control scheme mentioned in the above section. The main content of this section includes researching a case about the Ethereum network [28], and completing the writing and deployment of the access control contract through the online compiler Remix-IDE [29]. In addition, this paper conducts correctness and security analysis, and assumes possible attacks. Finally, this paper compares encryption efficiency with previous work.

4.1 Environment deployment

Remix is the main compilation tool used to deploy smart contracts. It can be compiled online or downloaded through the official website to compile. This article uses online compilation. In the creation of a smart contract, three stages are involved, writing, compiling, and declaring through the use of Solidity programming. The bytecode is generated by the Solidity real-time compiler, and all smart contracts must be uploaded to the blockchain in the form of bytecode. In order to announce smart contracts to the blockchain, an ether wallet is used. In order to simulate the user roles and their behavior in the method, ethereum nodes need to be created to simulate the behavior in the proposed method. The specific configuration of the machine is shown in Table 1:

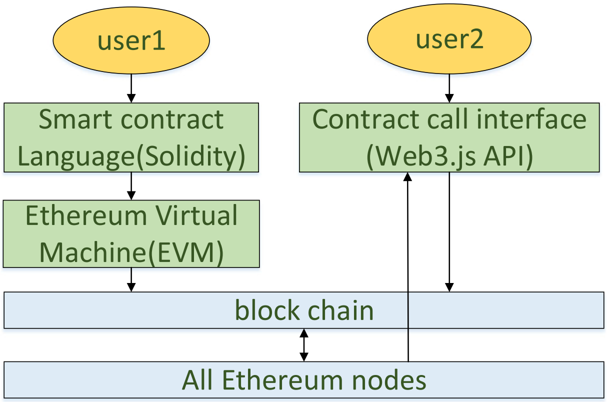

user1
user2
Smart contract Language(Solidity)
Contract call interface (Web3.js API)
Ethereum Virtual Machine(EVM)
block chain
All Ethereum nodes

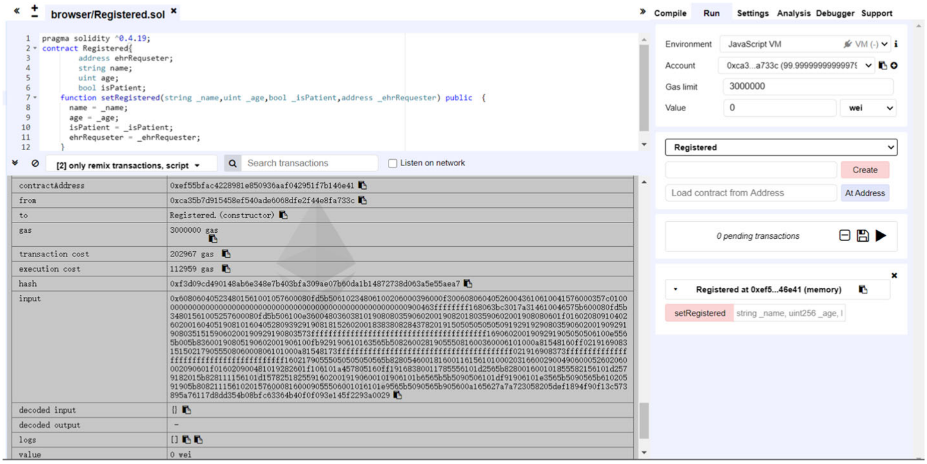
Fig. 7 Smart contract deployment and call flow chart

We need two contract accounts to complete the operation to the access control system. Among them, one account serves as the owner of the PMHR, and the other account serves as the requesting visitor of the PMHR. The two accounts each run a node and form a dedicated private blockchain network. Meanwhile, an Ethereum JavaScript API, i.e., Web3.js is used to communicate with the corresponding geth client through HTTP connections. The specific contract deployment and calling process is shown in Fig. 7:

4.2 Solidity deployment

Figure 8 shows the deployment of the user role registration contract on the Ethereum network. This method uses the online Remix-IDE compiler, and the upper left corner is the contract code written by Solidity. The upper right corner is the deployment tool and account information. In the lower right corner are all the function names and input boxes in the contract. The lower left corner is the output log of this contract, which includes information related to the deployment of smart contracts. The explanation of each symbol in the log is as follows:

1. **status.** It is "1" meaning that the transaction has been mined and executed successfully; otherwise, it is "0".
2. **transaction hash.** It is the hash of the transaction.
3. **from.** It is the address of the account from which the smart contract was commenced.
4. **to.** It is the address of the smart contract.
5. **gas.** It shows the amount of gas spent. In order to prevent malicious users from deploying contracts that run in an infinite loop, Ethereum requires users to pay for each step of the contract.
6. **transaction cost.** It is the amount of gas consumed.
7. **hash.** It is the hash of the smart contract.
8. **input.** It is the input value shown in hex.
9. **decoded input.** It depicts the decoded input.
10. **logs.** It is the transaction log.
11. **value.** It is the value in Wei in the smart contract.



```
pragma solidity ^0.4.19;
contract Registered{
        address ehrRequseter;
        string name;
        uint age;
        bool isPatient;
    function setRegistered(string _name,uint _age,bool _isPatient,address _ehrRequester) public  {
     name = _name;
     age = _age;
     isPatient = _isPatient;
     ehrRequseter = _ehrRequester;
    }
```

```
contractAddress    0xef55bfac4228981e850936aaf042951f7b146e41
from               0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to                 Registered.(constructor)
gas                3000000 gas
transaction cost   202967 gas
execution cost     112959 gas
hash               0xf3d09cd490148ab6e348e7b403bfa309ae07b60da1b14872738d063a5e55aea7
input              0x608060405234801561001057600080fd5b5061023480610020600039600f3006080604052600436106100415760003357c0100
                   00000000000000000000000000000000000000000000000000000900463ffffffff168063bc3017a314610046575b600080fd5b
                   348015610052576000 80fd5b506100e36004803603810190808035906020019082018035906020019080806 01f01602080910402
                   602001604051908101604052809392919081815260200183838082843782019150505050505091929192908035906020019092 91
                   908035151590602001909291908035 73ffffffffffffffffffffffffffffffffffffffff1690602001909291905050506100e556
                   5b005b8360019080519060200190610 0fb929190610163565b50826002819055508160036000 6101000a81548160ff0219169083
                   151502179055508060008 06101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffff
                   ffffffffffffffffffffffffffff160217905550505050505 65b828054600181600116156101000203166002900490600052602060
                   002090601f016020900481019282601f106101a457805160ff1916838001178555 56101d2565b82800160010185558215610 1d257
                   9182015b828111156101d157825182559160200191906001019061 01b6565b5b5090506101df91906101e3565b5090565b610205
                   91905b808211156102015760008160009055506001016101e9565b5090565b905600a165627a7a723058205def1894f90f13c573
                   895a76117d8dd354b08bfc63364b40f0f093e145f2293a0029
decoded input      {}
decoded output     -
logs               []
value              0 wei
```

Environment JavaScript VM VM (-)

Account 0xca3...a733c (99.999999999997§

Gas limit 3000000

Value 0 wei

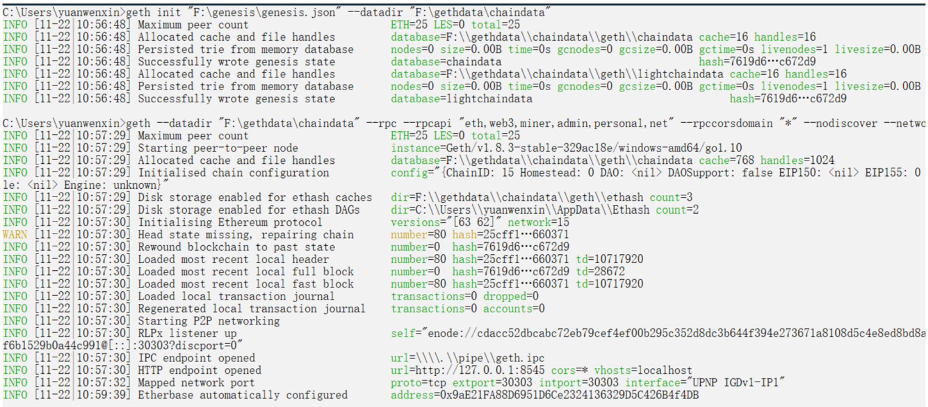
Registered

Load contract from Address

Registered at 0xef5...46e41 (memory)

setRegistered string _name, uint256 _age, l

Fig. 8 User registration smart contract



```
C:\Users\yuanwenxin>geth init "F:\genesis\genesis.json" --datadir "F:\gethdata\chaindata"
INFO [11-22|10:56:48] Maximum peer count                       ETH=25 LES=0 total=25
INFO [11-22|10:56:48] Allocated cache and file handles         database=F:\\gethdata\\chaindata\\geth\\chaindata cache=16 handles=16
INFO [11-22|10:56:48] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [11-22|10:56:48] Successfully wrote genesis state         database=chaindata                                 hash=7619d6…c672d9
INFO [11-22|10:56:48] Allocated cache and file handles         database=F:\\gethdata\\chaindata\\geth\\lightchaindata cache=16 handles=16
INFO [11-22|10:56:48] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [11-22|10:56:48] Successfully wrote genesis state         database=lightchaindata                                 hash=7619d6…c672d9

C:\Users\yuanwenxin>geth --datadir "F:\gethdata\chaindata" --rpc --rpcapi "eth,web3,miner,admin,personal,net" --rpccorsdomain "*" --nodiscover --netwo
INFO [11-22|10:57:29] Maximum peer count                       ETH=25 LES=0 total=25
INFO [11-22|10:57:29] Starting peer-to-peer node               instance=Geth/v1.8.3-stable-329ac18e/windows-amd64/go1.10
INFO [11-22|10:57:29] Allocated cache and file handles         database=F:\\gethdata\\chaindata\\geth\\chaindata cache=768 handles=1024
INFO [11-22|10:57:29] Initialised chain configuration          config="{ChainID: 15 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0
le: <nil> Engine: unknown}"
INFO [11-22|10:57:29] Disk storage enabled for ethash caches   dir=F:\\gethdata\\chaindata\\geth\\ethash count=3
INFO [11-22|10:57:29] Disk storage enabled for ethash DAGs     dir=C:\\Users\\yuanwenxin\\AppData\\Ethash count=2
INFO [11-22|10:57:30] Initialising Ethereum protocol           versions="[63 62]" network=15
WARN [11-22|10:57:30] Head state missing, repairing chain      number=80 hash=25cff1…660371
INFO [11-22|10:57:30] Rewound blockchain to past state         number=0  hash=7619d6…c672d9
INFO [11-22|10:57:30] Loaded most recent local header          number=80 hash=25cff1…660371 td=10717920
INFO [11-22|10:57:30] Loaded most recent local full block      number=0  hash=7619d6…c672d9 td=28672
INFO [11-22|10:57:30] Loaded most recent local fast block      number=80 hash=25cff1…660371 td=10717920
INFO [11-22|10:57:30] Loaded local transaction journal         transactions=0 dropped=0
INFO [11-22|10:57:30] Regenerated local transaction journal    transactions=0 accounts=0
INFO [11-22|10:57:30] Starting P2P networking
INFO [11-22|10:57:30] RLPx listener up                         self="enode://cdacc52dbcabc72eb79cef4ef00b295c352d8dc3b644f394e273671a8108d5c4e8ed8bd8a
f6b1529b0a44c991@[::]:30303?discport=0"
INFO [11-22|10:57:30] IPC endpoint opened                      url=\\\\.\\pipe\\geth.ipc
INFO [11-22|10:57:30] HTTP endpoint opened                     url=http://127.0.0.1:8545 cors=* vhosts=localhost
INFO [11-22|10:57:32] Mapped network port                      proto=tcp extport=30303 intport=30303 interface="UPNP IGDv1-IP1"
INFO [11-22|10:59:39] Etherbase automatically configured       address=0x9aE21FA88D6951D6Ce2324136329D5C426B4f4DB
```

Fig. 9 Construction of contract environment

Figure 9 shows the creation of the genesis block and the construction of the environment for running smart contracts.

After setting up the operating environment of the contract, web3 connects to the Ethereum nodes. After web3 is successfully connected, operations such as deploying smart contracts and calling functions in smart contracts can be performed. Figure 10 shows the creation and deployment of the contract, and Fig. 11 shows the result after calling the contract. The watermark in the picture is a built-in effect of the online Remix-IDE compiler.

4.3 Analysis

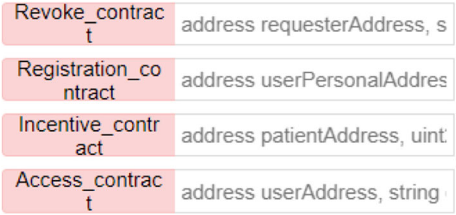
4.3.1 Correctness analysis

The correctness analysis refers to the verification of the correctness of the ciphertext involved in this scheme. By operating the algorithm proposed in this program in the steps of encryption first and then decryption, it is proved that the ciphertext encrypted by the patient and the plaintext by the visitor decryption are the same as the original data.

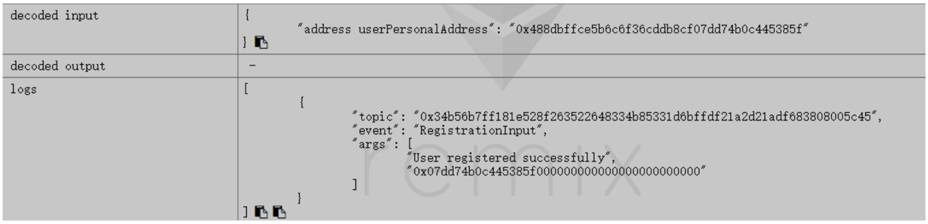
There are three ciphertexts in this scheme, including the ciphertext c encrypted by the patient through the AES encryption algorithm, the ciphertext c_1 after the patient encrypts the AES key k through the RSA encryption algorithm, and the ciphertext re-encrypted by the cloud server c_2.

First verify the correctness of the first ciphertext c : the PMHR m is encrypted by the patient p_i to obtain the ciphertext $c = \mathsf{AES.encrypt}(m, k)$, it means that PMHR is encrypted by the AES encryption algorithm, the input parameters are PMHR m and AES key k. The

Fig. 10 Creation and deployment of contract



Revoke_contract address requesterAddress, s
Registration_contract address userPersonalAddres
Incentive_contract address patientAddress, uint
Access_contract address userAddress, string



```
decoded input    {
                         "address userPersonalAddress": "0x488dbffce5b6c6f36cddb8cf07dd74b0c445385f"
                 }
decoded output   -
logs             [
                         {
                                 "topic": "0x34b56b7ff181e528f263522648334b85331d6bffdf21a2d21adf683808005c45",
                                 "event": "RegistrationInput",
                                 "args": [
                                         "User registered successfully",
                                         "0x07dd74b0c445385f000000000000000000000000"
                                 ]
                         }
                 ]
```

Fig. 11 The result of calling the contract

second ciphertext is the ciphertext $c_1 = \mathsf{rsa.encrypt}(k, pk_i)$, it is encrypted by the patient p_i through RSA encryption algorithm. The input parameters are AES key k and public key pk_i of the patient p_i. Input the ciphertext c_1 and the patient's private key sk_i through the RSA encryption algorithm, calculate $\mathsf{rsa.decrypt}(c_1, sk_i)$, obtain the plaintext k' of c_1, and then input the ciphertext c and k' through the AES encryption algorithm, calculate $\mathsf{AES.decrypt}(c, k')$, finally obtain the plaintext m'. After verification, it can be known that the decrypted plaintext m' is equal to the plaintext PMHR m provided by the patient p_i, which is correct.

The third ciphertext c_2 is the ciphertext after re-encryption. Verify the correctness of the re-encrypted ciphertext c_2: It is known that the agent re-encryption key $K_{A,B}$. First, the ciphertext c_1 is re-encrypted, input the parameters $K_{A,B}$ and the ciphertext c_1, calculate $c_2 = \mathsf{reEncrypt}(K_{A,B}, c_1)$. Then calculate $\mathsf{rsa.decrypt}(c_2, sk_j)$, input the private key sk_j and ciphertext c_2 of the visitor a_j, and get the plaintext k'. After verification, it can be known that the decrypted plaintext k' is equal to the AES key k, and it can be determined that the proxy re-encryption proposed in this scheme is correct.

4.3.2 Security analysis

This section introduces the security of PMHR stored in cloud server and blockchain in detail. First, it explains how the PMHR stored in the cloud server CS, then assume the security of PMHR when the cloud server acts as a man-in-the-middle attack. Finally it explains how the PMHR stored in blockchain uses the characteristics of the blockchain to ensure the security.

In this solution, the PMHR is first generated through the AES encryption algorithm to generate an encrypted file c, then the patient generates his own key pair sk_i and pk_i through the RSA encryption algorithm. The patient's public key pk_i is used to encrypt the AES key k, then the patient p_i obtains a ciphertext c_1. The two ciphertexts c and c_1 are passed through the patient's medical institution M, then they are sent to the cloud server CS. Since the attacker does not have the patient's private key sk_i, the attacker cannot obtain the AES key k through decryption and thus cannot obtain the PMHR of the patient p_i. Assuming that there is an attack situation, the specific content is as follows:

A man-in-the-middle attack is an attack mode against public-key encryption algorithms that can tamper with encrypted content. According to the scheme of this article, suppose that the middleman is the cloud server CS, the communicating party is the patient P and the medical company M. In the previous work, first M sends his public key to P through CS, and P encrypts his PMHR through $M's$ public key. If $P's$ access permission is obtained, the ciphertext is sent to M through CS. M decrypts with his own private key to obtain the PMHR. But if CS intercepts $M's$ public key during the transmission process and sends his

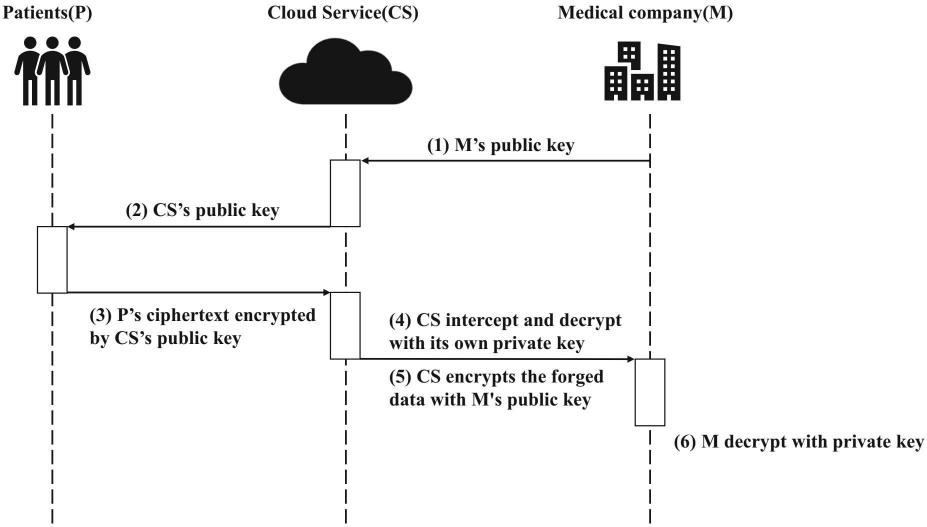

Patients(P)
Cloud Service(CS)
Medical company(M)
(1) M's public key
(2) CS's public key
(3) P's ciphertext encrypted by CS's public key
(4) CS intercept and decrypt with its own private key
(5) CS encrypts the forged data with M's public key
(6) M decrypt with private key

Fig. 12 Block diagram of the man-in-the-middle attack process

own public key to P, CS can obtain $P's$ PMHR, or even tamper with $P's$ PMHR, and then pass $M's$ public key send it to M after encryption. The specific attack mode is shown in Fig. 12.

In this scheme, since $M's$ public key is sent to P directly after he obtains $P's$ access rights, it does not need to be transmitted by CS, so CS cannot obtain $M's$ public key, and CS only provides encrypted data transmission services. It is impossible to tamper or steal the ciphertext, so the attack is unsuccessful.

This solution uses the blockchain to store the hash value of the PMHR, which can ensure the authenticity and integrity of the data. Because the blockchain has the characteristics of tamper-proofing, once data is stored in the blockchain, all nodes on the blockchain network will have a backup of the data. If an attacker tampered with the data in some nodes, it is also the erroneous data will be discovered and corrected by other nodes, which greatly improves the security and privacy of PMHR. Moreover, the attack on the hash value stored on the blockchain will not affect the original medical data.

Table 2 shows the comparison of data security between this scheme and [12] in the four stages of data collection, data tracking, data storage, and data sharing. According to this table, it can be concluded that this scheme adopts a proxy re-encryption scheme to transmit data, which ensures the safety of patient data during the entire scheme.

Table 2 Comparison of data security at each stage

	Gan et al. [12]	This paper
Data Collecting	√	√
Data Tracing	√	√
Data Storage	×	√
Data Sharing	×	√

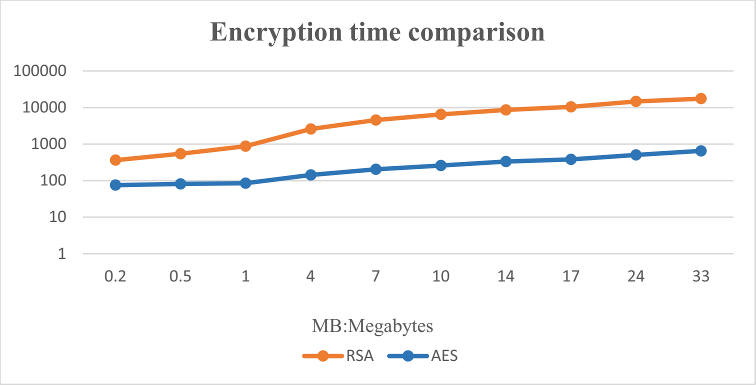


Fig. 13 Encryption time comparison chart

4.3.3 Performance and efficiency analysis

In order to further evaluate the efficiency of this scheme, this part conducts a simulation test on the improved proxy re-encryption algorithm in this scheme. The equipment configuration evaluated was performed on a notebook computer with AMD Ryzen7 4800H, CPU 2.90GHz and 16GB.

First, perform encryption and decryption operations through RSA and AES respectively. Figures 13 and 14 are time comparison diagrams of encryption and decryption through RSA and AES respectively. From the figures, the following conclusions can be drawn: Compared with the AES encryption algorithm, the encryption efficiency of the RSA encryption algorithm is not obvious when encrypting small data, but as the data increases, the time required for the RSA encryption algorithm to encrypt is much longer than that of the AES encryption. algorithm. The comparison of time cost in decryption is more obvious, especially when decrypting large data files, the time consumed by RSA encryption algorithm decryption is very long. This is because the length of the RSA algorithm public key and private key varies with the size of the encrypted data. Decided. Although the AES encryption algorithm is very fast compared to RSA when encrypting and decrypting data files, the separate AES

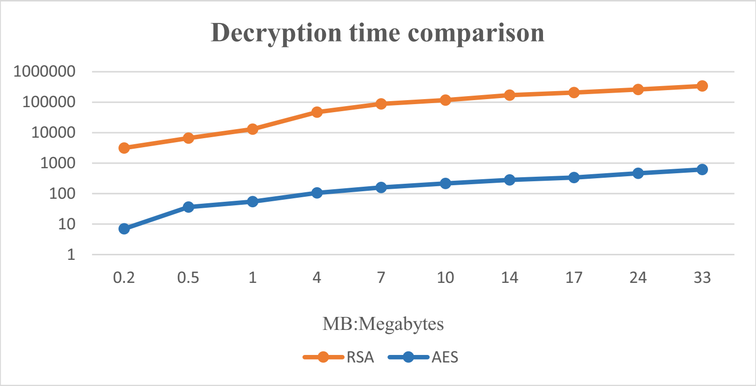


Fig. 14 Decryption time comparison chart

encryption algorithm is symmetric encryption, and the generated key needs to be transmitted through the cloud server, resulting in a certain security risk for the key. Therefore, proxy re-encryption operations cannot be performed, and the security of patient data cannot be guaranteed.

The AES and RSA hybrid encryption proposed in this solution uses the AES encryption algorithm to encrypt PMHR, and the RSA encryption algorithm encrypts the key of the AES encryption algorithm. Although the effect of this hybrid encryption method is not obvious when the data is small, as the data increases, the hybrid encryption method proposed in this solution not only ensures the security of the patient's private data during the entire transmission process, but also improves the entire encryption. The efficiency of the algorithm. In addition, since the patient's medical and health records include big data files such as images, AES and RSA hybrid encryption is more suitable for medical and health systems. Figure 15 shows the total computational cost (encryption cost + proxy re-encryption cost + decryption cost) of the proposed AES and RSA hybrid encryption algorithm and other related schemes [9, 26, 31] as a function of data.

In [9, 26, 31], it can be seen from the figure that the scheme of proxy re-encryption is adopted to encrypt the plaintext. The difference from the solution proposed in this article is that the specific patient encryption and visitor decryption operations use a public key encryption algorithm. This article proposes an improved proxy re-encryption solution, it uses a hybrid encryption algorithm to encrypt and decrypt PMHR. It shortens the time for patients to encrypt PMHR and visitors to decrypt it. From an overall point of view, although there is no obvious advantage in time cost compared with other programs when the PMHR is small, as the PMHR gradually increases, it is obvious that this program consumes more time than other programs in the process. There are relatively few other programs, and the larger the data, the more obvious the contrast. Therefore, this solution consumes less time, it also has higher security and avoids the possibility of collusion attacks on cloud servers. This solution obviously has certain advantages and practicability.

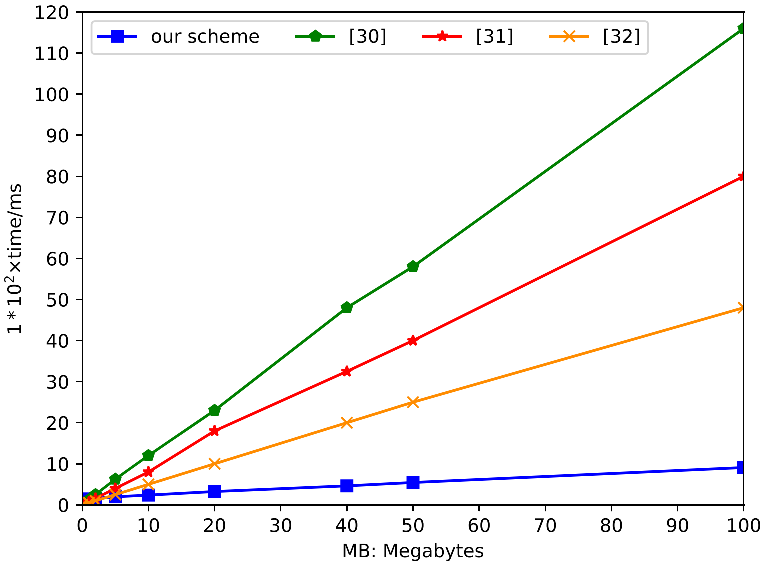


Fig. 15 Total calculated cost comparison chart

Table 3 Comparison with similar systems

	Gan et al. [12]	Eltayieb et al. [9]	Wang et al. [31]	This paper
Decentralization	√	√	√	√
Smart contract	√	×	×	√
Incentive function	√	×	×	√
Resist server attacks	×	√	√	√

In addition, the expected functions include the following: (1) Decentralization, (2) Smart contract, (3) Incentive function, (4) Resist server attacks. The results are summarized in Table 3. As can be seen from this comparison, the proposed scheme in this paper can simultaneously meet all the requirements.

5 Conclusions

In this paper, we propose a blockchain PMHR access control scheme based on improved proxy re-encryption to research the application of blockchain in today's medical and health systems. In the previous works, any third-party visits must be confirmed by the patient himself, which increased the pressure on the patient and reduced the efficiency of the medical institutions. In this solution, the blockchain-based access control solution only allows medical institutions and pharmaceutical companies to apply for access. This solution completes data transmission through AES and RSA hybrid encryption, it improves the efficiency of the entire access system while protecting the rights of patients. Finally, a case study was conducted on the access control system through the ethereum network, and the efficiency of various encryption algorithms is compared through simulation, which demonstrated the feasibility and practicability of the scheme.

The significance of this scheme is to realize efficient data sharing while protecting the PMHR. Although this program gives patients control over their own data, it will not affect the diagnosis of medical institutions and research in the medical field. On the contrary, this program encourages patients to actively share PMHR so as to contribute to research in the medical field [6, 19].

Funding This work was funded by Shandong Provincial Natural Science Foundation (No. ZR2021MF050), the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (No. 18YJAZH110), and the National Statistics Science Project (No. 2021LY082).

The authors have no relevant financial or non-financial interests to disclose.

References

1. Alok N, Krishan K, Chauhan P (2021) Deep learning-based image classifier for malaria cell detection. Mach Learn Healthcare Appl:187–197
2. Amofa S, Sifah EB, Kwame O-B, Abla S, Xia Q, Gee JC, Gao J (2018) A blockchain-based architecture framework for secure sharing of personal health data. In: 2018 IEEE 20th international conference on e-health networking, applications and services (Healthcom). IEEE, pp 1–6
3. Ansari H, Vijayvergia A, Kumar K (2018) Dcr-hmm: depression detection based on content rating using hidden markov model. In: 2018 conference on information and communication technology (CICT), IEEE. pp 1–6

4. Chen Y, Ding S, Xu Z, Zheng H, Yang S (2019) Blockchain-based medical records secure storage and medical service framework. J Med Syst 43(1):1–9
5. Chen HS, Jarrell JT, Carpenter KA, Cohen DS, Huang X (2019) Blockchain in healthcare: a patient-centered model. Biomed J Sci Tech Res 20(3):15017
6. Dabral I, Singh M, Kumar K (2019) Cancer detection using convolutional neural network. In: International conference on deep learning, artificial intelligence and robotics. Springer, pp 290–298
7. Darbari A, Kumar K, Darbari S, Patil PL (2021) Requirement of artificial intelligence technology awareness for thoracic surgeons. The Cardiothoracic Surgeon 29(1):1–10
8. Durao F, Carvalho JFS, Fonseka A, Garcia VC (2014) A systematic review on cloud computing. J Supercomput 68(3):1321–1346
9. Eltayieb N, Sun L, Wang K, Li F (2019) A certificateless proxy re-encryption scheme for cloud-based blockchain. In: International conference on frontiers in cyber security. Springer, pp 293–307
10. Esposito C, De Santis A, Tortora G, Chang H, Choo K-KR (2018) Blockchain: a panacea for healthcare cloud-based data security and privacy. IEEE Cloud Comput 5(1):31–37
11. Eyal I (2017) Blockchain technology: transforming libertarian cryptocurrency dreams to finance and banking realities. Computer 50(9):38–49
12. Gan C, Saini A, Zhu Q, Xiang Y, Zhang Z (2020) Blockchain-based access control scheme with incentive mechanism for ehealth systems: patient as supervisor. Multimed Tools Appl:1–17
13. Gordon WJ, Catalini C (2018) Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability. Computat Structural Biotechnol J 16:224–230
14. Guo H, Li W, Nejad M, Shen C-C (2019) Access control for electronic health records with hybrid blockchain-edge architecture. In: 2019 IEEE international conference on blockchain Blockchain, IEEE, pp 44–51
15. Jin H, Luo Y, Li P, Mathew J (2019) A review of secure and privacy-preserving medical data sharing. IEEE Access 7:61656–61669
16. Karame G, Capkun S (2018) Blockchain security and privacy. IEEE Secur Privacy 16(04):11–12
17. Khezr S, Moniruzzaman M, Yassine A, Benlamri R (2019) Blockchain technology in healthcare: a comprehensive review and directions for future research. Appl Sci 9(9):1736
18. Kumar A, Singh N, Kumar P, Vijayvergia A, Kumar K (2017) A novel superpixel based color spatial feature for salient object detection. In: 2017 Conference on information and communication technology (CICT). IEEE, pp 1–5
19. Kumari S, singh M, Kumar K (2019) Prediction of liver disease using grouping of machine learning classifiers. In: International conference on deep learning, artificial intelligence and robotics. Springer, pp 339–349
20. Li H, Yang Y, Dai Y, Yu S, Xiang Y (2017) Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Trans Cloud Comput 8(2):484–494
21. Liu X, Xia Y, Yang W, Yang F (2018) Secure and efficient querying over personal health records in cloud computing. Neurocomputing 274:99–105
22. Mansfield-Devine S (2017) Beyond bitcoin: using blockchain technology to provide assurance in the commercial world. Comput Fraud Secur 2017(5):14–18
23. Mikula T, Jacobsen RH (2018) Identity and access management with blockchain in electronic healthcare records. In: 2018 21st euromicro conference on digital system design (DSD). IEEE, pp 699–706
24. Negi A, Kumar K, Chauhan P (2021) Deep neural network-based multi-class image classification for plant diseases. Agricultural Inform Autom Using IoT Mach Learn:117–129
25. Negi A, Kumar K, Chauhan P, Rajput R (2021) Deep neural architecture for face mask detection on simulated masked face dataset against covid-19 pandemic. In: 2021 International conference on computing, communication, and intelligent systems (ICCCIS). IEEE, pp 595–600
26. Noh S-W, Park Y, Sur C, Shin S-U, Rhee K-H (2017) Blockchain-based user-centric records management system. Int J Control Autom 10(11):133–144
27. Omar AA, Rahman MS, Basu A, Kiyomoto S (2017) Medibchain: a blockchain based privacy preserving platform for healthcare data. In: International conference on security, privacy and anonymity in computation, communication and storage
28. Ranganthan VP, Dantu R, Paul A, Mears P, Morozov K (2018) A decentralized marketplace application on the ethereum blockchain. In: 2018 IEEE 4th international conference on collaboration and internet computing (CIC). IEEE, pp 90–97
29. Remix-IDE. http://remix.ethereum.org/. Accessed Jan 2019
30. Tanwar S, Parekh K, Evans R (2020) Blockchain-based electronic healthcare record system for healthcare 4.0 applications. J Inf Secur Appl 50:102407
31. Wang Z, Tian Y, Zhu J (2018) Data sharing and tracing scheme based on blockchain. In: 2018 8th International conference on logistics, informatics and service sciences (LISS). IEEE, pp 1–6

32. Wang Y, Zhang A, Zhang P, Wang H (2019) Cloud-assisted ehr sharing with security and privacy preservation via consortium blockchain. IEEE Access 7:136704–136719
33. Zhu T-L, Chen T-H (2021) A patient-centric key management protocol for healthcare information system based on blockchain. In: 2021 IEEE conference on dependable and secure computing (DSC). IEEE, pp 1–5

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Wen-Xin Yuan received his B.E. degree in automation, from College of Electrical Engineering and Automation, Northern University for Nationalities, P. R. China, in 2018. He is now pursuing his master's degree in College of Electronic and Information Engineering, Shandong University of Science and Technology. His research interest is blockchain-based medical and health record sharing and protection.

Bin Yan received the B.S. degree in applied physics from Qingdao University, P. R. China, in 1996 and the M.S. degree in electrical engineering from Harbin Institute of Technology, P. R. China, in 2002 and Ph. D degree in electrical engineering from Harbin Institute of Technology, P. R, China, in 2007. From 1996-1999, he was an engineer in Goma Company Group. From 2007-2012, he was a lecturer in Shandong University of Science and Technology. From 2015-2016, he was a visiting scholar in Deakin University, Australia. From 2013-2018, he was an associate professor in Shandong University of Science and Technology. Since 2019, he has been a full professor in Shandong University of Science and Technology. His research interests include statistical signal processing, multimedia signal processing and security.

Wen Li received her master's degree from Shandong University of Science and Technology in 2014. She is now with the Confidentiality Administration Bureau of Ji-Ning, P. R. China. Her research interests include visual cryptography and blockchain security.

Liu-Yao Hao received her B.E. degree in electronic information engineering, from the College of Information Science and Engineering, Wanfang College of Science and Technology, Henan Polytechnic University, 2017, P. R. China. She received her master's degree from Shandong University of Science and Technology in 2020. She is now with the China Mobile Communications Research Institute. Her research interests include blockchain applications, multimedia security, and communication security.

Hong-Mei Yang was born in Shandong province, P. R. China in 1969. She received her Ph.D. degree from Shandong University of Science and Technology in 2009. She is now with the College of Computer Science and Engineering, Shandong University of Science and Technology. Her research interests include digital watermarking and image quality evaluation.