symmetry

MDPI

Article

Privacy-Preserving Medical Data-Sharing System with Symmetric Encryption Based on Blockchain

Mingqi Hu, Yanli Ren * and Cien Chen

School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China; humingqi@shu.edu.cn (M.H.); cce97@shu.edu.cn (C.C.)

* Correspondence: renyanli@shu.edu.cn

Abstract: Nowadays, data between hospitals are usually not interoperable, which brings great inconvenience to medical data sharing and patients' medical treatment. In addition, patients do not want their medical data to be leaked during the sharing process. Researchers have employed blockchain to build data-sharing systems to address these issues. However, current systems do not restrict the power of participants, nor do they prevent visitors from sharing the obtained data to unauthorized parties. To address these issues, we propose a private data-sharing system with symmetric encryption for the medical industry that implements power restriction and access control, and prevents the leakage of private data. To be specific, firstly, symmetric encryption algorithm is utilized to encrypt medical data to protect the privacy of data owner. Secondly, our proposed system is built on a new blockchain framework, in which only visitors with permission can access the medical data. Thirdly, we employ chameleon signature to prevent visitors from sharing data with other parties without permission. Finally, we make the power of participants in the system revocable to prevent them from abusing their power. Our proposed system has been proven to be secure through security analysis and can protect the privacy of patients. In addition, the experimental results show that our system has excellent performance in terms of time overhead compared to other systems.

Keywords: blockchain; medical data sharing; power supervision; access control

check for updates

Citation: Hu, M.; Ren, Y.; Chen, C. Privacy-Preserving Medical Data-Sharing System with Symmetric Encryption Based on Blockchain. *Symmetry* **2023**, *15*, 1010. https://doi.org/10.3390/sym15051010

Academic Editor: Lorentz Jäntschi

Received: 3 April 2023
Revised: 27 April 2023
Accepted: 28 April 2023
Published: 30 April 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the improvement of people's living quality, health problems are receiving greater attention [1]. Effectively using patients' medical data is crucial for improving people's health, which requires hospitals to store and share medical data efficiently and securely. However, hospitals usually store patients' data locally and do not share them with others. This lack of circulation of medical data causes significant inconvenience to patients and some medical research. When patients seek treatment at a different hospital, if their previous medical data cannot be accessed, they may need to undergo repeated examinations, resulting in wasted time and money. More seriously, it may cause medical accidents. Medical data are also a valuable resource for medical research [2], and a significant amount of medical data need to be used when conducting medical research. Therefore, the medical data in only a hospital's database are not enough. Some cases of research value are difficult to aggregate together without data circulation. However, the hospital only stores the data instead of sharing them, which leads to a waste of medical data. In addition, ensuring the security of medical data is also a significant challenge that cannot be ignored. In the medical field, there are many incidents of data breaches every year [3], such as data tampering, data leaks, and unauthorized access. The increasing digitization of the medical field also demands that people take this issue seriously. If the data cannot be guaranteed to be secure, the interests of patients will be compromised [4], which is unacceptable.

In recent years, blockchain [5] has been utilized for medical data sharing [6], which is a decentralized distributed ledger. The transactions in the ledger cannot be tampered

with, and each client in the system has the complete transaction of the ledger. Data in the blockchain are stored in blocks, which are added to the blockchain in chronological order. The new block is generated by the consensus protocol and connected to the blockchain chronologically through hash value. Due to its immutability, security [7], anonymity and data integrity [8], blockchain has been widely used in electronic currency, data storage, data integrity verification [9], and data sharing. In the future, blockchain will play an irreplaceable role in various fields, such as federated learning [10–12], finance [13] and medicine, especially in the medical field [14,15].

In the medical field, blockchain can play a crucial role in improving data security and interoperability [16]. According to [17], people require data to be secure, but this cannot always be achieved. We have found that blockchain can solve this problem to some extent. Firstly, there is often a lack of secure and reliable methods for storing medical data in the medical field. Blockchain technology provides a decentralized database that can securely store medical data to prevent malicious tampering. Secondly, blockchain can be leveraged to enhance data interoperability. Since information systems in various hospitals are often built using incompatible technologies, data sharing among hospitals has become a challenging task. The emergence of blockchain provides hospitals with a unified, decentralized database, enabling hospitals to access and share medical data more easily. Additionally, quantum communication [18–21] is also a promising solution to address the aforementioned issues, but it is challenging to conduct experiments due to hardware limitations. Therefore, due to blockchain's unique advantages in the secure sharing of medical data [22], it can be used to design a secure and efficient data-sharing system. However, designing such a system still faces some security challenges. Firstly, the issue of how to restrict the power of data managers needs to be solved [23]. Most systems provide data managers with great power, but data owners have no corresponding power to restrict them. The second challenge is preventing shared data from being leaked to unauthorized parties. Once the data manager shares the data, he also loses control over them. In this situation, visitor can freely share the data with others, which seriously compromises the interests of data owner. The last challenge is access control, which means that only authorized visitors can access the data. In recent years, several systems have been proposed to address the aforementioned challenges, which will be introduced briefly as follows.

Related work. In order to realize the function of access control, Rahulamathavan et al. [24] utilized attributed-based encryption (ABE) [25] to protect the privacy of data by storing the encrypted data on the blockchain, resulting in high storage overhead. The system implements access control functions and protects user privacy to some extent. However, the system does not restrict the power of participants, nor does it consider the situation where the data are shared with unauthorized third parties by the visitor after obtaining them. Additionally, since the encrypted data are stored on the blockchain, this system incurs a high data storage overhead. In 2021, Qi et al. proposed a sharing system [26] that supports data compression called Cpds, which encrypts the compressed data by symmetry encryption and submits the ciphertext to the blockchain. In addition, Cpds implements access control by encrypting the symmetry key and sending the ciphertext to the blockchain, reducing the storage overhead by compressing data before encryption. However, visitor can share the obtained data with unauthorized parties in Cpds, which greatly infringes upon the interests of the patients. At the same time, the power of the system participants is so great that they can even change previously generated data. Therefore, an effective algorithm is needed to limit the power of nodes. Du et al. proposed a medical data-sharing system [27] based on blockchain. The data sharer stores the data digest in the platform's database while the complete data are stored locally. Authorized access by the sharer is required to obtain corresponding data. However, the system does not solve the problem of data misuse after sharing, and there is no specific algorithm for authorizing operations. In addition, as the hospital can maliciously send patients' medical data to unauthorized parties, the power restriction function is also not implemented. In order to address the issue

of data misuse, Wang et al. [28] and Xiao et al. [29] designed data-sharing systems based on blockchain that utilize trusted execution environments (TEEs), such as Intel SGX. In these two systems, an algorithm, negotiated in advance, is used to process the original data in TEE. Afterwards, the calculation result is sent to the visitor without revealing the original data. Therefore, these two systems solve the problem of data misusing. However, these systems still do not limit the power of the participants. Moreover, as visitors may apply the data in areas such as machine learning that require the original data, it is inappropriate to only send the computation results in such cases. Nguyen et al. proposed the new distributed medical network architecture BEdgeHealth [30] to transfer and share data, which utilizes smart contracts to enable data sharing without a third party. However, patients are unable to control their own data in this system. In addition, in 2022, Wu et al. proposed a blockchain-based smart healthcare system [31] for medical data exchanging and sharing. The system deploys smart contracts to meet the requirements of access control and data sharing, which improves the efficiency of the system. However, similar to BEdgeHealth, this system does not restrict the power of participants nor consider the issue of data misuse. In addition, Ref. [32] provides a secure data-sharing scheme for different levels of visitors. The scheme employs a ciphertext-based attribute encryption algorithm, allowing for more fine-grained access control. However, the power possessed by the administrator of the system attributes is too great and cannot be limited such that it can ignore or incorrectly respond to user's requests. In addition, the access control function is still not implemented in this system.

Overall, as shown in Table 1, although ABE is utilized for implementing access control in [24,26], it does not limit the permission of participants in the systems, nor does it design an algorithm to prevent the shared data from being leaked to unauthorized parties. The protection of data privacy is realized in the system of [27], but the remaining functions in the table are not realized. Systems of [28,29] use intel SGX [33] technology to ensure that the shared data are not leaked and that they have access control function, but they lack permission revocation. Systems of [30–32] utilize smart contracts to protect user privacy and implement access control. However, neither of these systems implemented functionalities permission revocation and use control.

Table 1. The comparison of protocols.

System	Access Control	Revocation of Power	Use Control	Data Privacy
Pbee [24]	✔	✘	✘	✔
Cpds [26]	✔	✘	✘	✔
Obms [27]	✘	✘	✘	✔
Spds [28]	✔	✘	✔	✔
PG [29]	✔	✘	✔	✔
BEdgeHealth [30]	✔	✘	✘	✔
Bshs [31]	✔	✘	✘	✔
SDSM [32]	✔	✘	✘	✔
Ours	✔	✔	✔	✔

To sum up, the current blockchain-based data-sharing systems still face several challenges that must be addressed. Firstly, data owners do not want unauthorized parties to obtain their data. Secondly, honest parties in the system hope that the power of malicious parties can be limited. Thirdly, after sharing the data with the visitors, the data owners do not want the data to be shared with a third party without permission by the visitor. Finally, data privacy should be protected [34]. Unfortunately, existing blockchain-based data-sharing systems cannot overcome these challenges simultaneously.

In this article, as shown in Table 1, a blockchain-based secure medical data-sharing system is proposed to meet these challenges. Specifically, a blockchain-based system is designed to securely share medical data. The data visitor is only eligible to access the data with the consent of the data owner. A permission revocation algorithm is also designed to

limit the power of data managers and prevent the abuse of power that could compromise patients' privacy. In addition, we also use the chameleon signature algorithm to prevent shared data from being leaked to unauthorized parties. In summary, our article has the following contributions:

1. A medical data-sharing system is proposed based on blockchain for secure data sharing between hospitals. This system only stores simple data records on the blockchain, while the complete data are encrypted and stored in the application platform.
2. A verification system is also proposed based on chameleon hash with revocable trapdoor, enabling patients to revoke the hospital's right of managing data and allowing hospitals to revoke the application platform's right of signing medical data.
3. The proposed system can prevent the misuse of shared data based on chameleon signature. By designing this algorithm, we solve the problem of data misuse, which prevents data that have been shared from being leaked to other parties who do not have permission to access them.

2. Preliminaries

The key technologies used in our system are introduced briefly in this section, including blockchain and chameleon hash.

2.1. Blockchain

Blockchain is a chained data structure that groups blocks of data together in chronological order, which is shared and maintained by all peers in a distributed system. Blocks in the blockchain are generated through a consensus protocol and appended to the blockchain using hash function. Whenever a new block is generated, it is broadcast to all peers in the system. Peers then connect the new block to the end of the blockchain through a hash function when they receive it. As shown in Figure 1, blocks are connected in such a way that each block stores the hash value of the previous block.

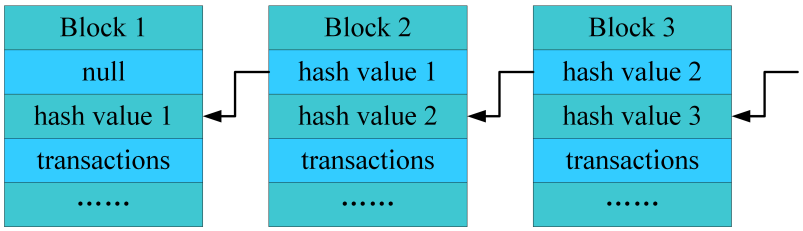


Figure 1. The chain structure of the blockchain.

Since the blocks in the blockchain are connected in this way, if the data in the previous block are tampered with, the blockchain will be broken. If the tamperer wants the blocks to remain connected after tampering, he needs to recalculate the hash value of all subsequent blocks through the proof-of-work algorithm. This requires the tamperer to have more computing power than the sum of the other nodes in the system, which is almost impossible. Therefore, the data in the blockchain cannot be tampered with.

2.2. Chameleon Hash

In the chameleon hash function, the party that calculates the chameleon hash value holds both the chameleon public key pk_{CH} and the chameleon private key (trapdoor) sk_{CH}. When calculating the chameleon hash value h, he needs to use pk_{CH} for the calculation. Unlike hash function $H()$, the chameleon hash calculator can use the trapdoor to compute the collision between the origin message m and new message m', resulting in the new chameleon hash value h' calculated from m' equal to h. Therefore, we utilize the chameleon hash function to allow the data owner to restrict the power of other participants. In addition, compared to the existing chameleon hash functions [35,36], our proposed chameleon hash function based on elliptic curves has better performance in terms of time overhead. The

parts of the chameleon hash ($Setup_{CH}$, $KeyGen_{CH}$, $Hash_{CH}$, $Adapt_{CH}$, and $Verify_{CH}$) are described as follows [37]:

1. $Setup_{CH}(1^{\lambda}) \rightarrow pp$: On inputting a security parameter λ, the algorithm outputs a system parameter pp.
2. $KeyGen_{CH}(pp) \rightarrow (pk_{CH}, sk_{CH})$: On inputting system parameter pp, the algorithm outputs a pair of private and public key (sk_{CH}, pk_{CH}).
3. $Hash_{CH}(pk_{CH}, m) \rightarrow (h, r)$: On inputting a public key pk_{CH} and a message m, the algorithm outputs chameleon randomness r and chameleon hash value h.
4. $Adapt_{CH}(sk_{CH}, (h, r, m), m') \rightarrow r'$: On inputting a private key sk_{CH}, a chameleon hash value h, a chameleon randomness r, a message m and a new message m', the algorithm outputs a new chameleon randomness r'.
5. $Verify_{CH}(h, pk_{CH}, m, r) \rightarrow \{0, 1\}$: On inputting a chameleon hash value h, a public key pk_{CH}, a message m and a chameleon randomness r, the algorithm outputs 1 if (h, r, m) is valid and otherwise outputs 0.

Three properties of secure chameleon hashes are given in [38]: correct, indistinguishable and collision resistance. However, in our system, it is not necessary to hide the computation of collisions, so indistinguishability is not applicable to our system.

Definition 1 (Secure Chameleon Hashes)**.** *If for all $\lambda \in \mathbb{N}$, m and $m' \in \mathcal{M}$, there is $Verify_{CH}(h, pk_{CH}, m, r) = 1$ and $Verify_{CH}(h, pk_{CH}, m', r') = 1$ for any $Setup_{CH}(1^{\lambda}) \rightarrow pp$, $KeyGen_{CH}(pp) \rightarrow (pk_{CH}, sk_{CH})$, $Hash_{CH}(pk_{CH}, m) \rightarrow (h, r)$ and $Adapt_{CH}(sk_{CH}, (h, r, m), m') \rightarrow r'$, the chameleon hash can be considered to satisfy correctness. If an efficient adversary without the chameleon private key sk_{CH} cannot compute collisions for any message, the chameleon hash can be considered to satisfy collision resistance. If a chameleon hash satisfies both correctness and collision resistance, it can be considered secure.*

3. Chameleon Hash with Revocable Trapdoor

Our goal is to design a secure medical data-sharing system in which the participants' rights can be revoked. To achieve this, we consider using a chameleon hash function with a revocable trapdoor. We first propose a chameleon hash function based on the elliptic curve group to improve the efficiency of the operation, and then give the specific construction of the chameleon hash with revocable trapdoor.

3.1. Chameleon Hash Based on Elliptic Curve Group

A chameleon hash based on elliptic curve group consists of five algorithms ($Setup_{CH}$, $Keygen_{CH}$, $Hash_{CH}$, $Adapt_{CH}$, and $Verify_{CH}$):

1. $Setup_{CH}(1^{\lambda}) \rightarrow pp$: Let G be the generator point of the elliptic curve group E, and the smallest n that satisfies $nG = O$ is a very large prime number, where O is the infinity point on the elliptic curve. The algorithm outputs the system parameter $pp = (E, G, n)$.
2. $KeyGen_{CH}(pp) \rightarrow (pk_{CH}, sk_{CH})$: Choose an integer $x(x < n)$ as the trapdoor and compute public key $Y = xG$. Then $pk_{CH} = Y$, $sk_{CH} = x$.
3. $Hash_{CH}(pk_{CH}, m) \rightarrow (h_{CH}, r)$: Randomly choose $r = (R, s)$, where R is a random point on the elliptic curve group and $s \in Z_n^*$. Compute $h = H(m)$ and $Q = (Q_x, Q_y) = R + hY + sG$. Then the chameleon hash value $h_{CH} = Q$.
4. $Adapt_{CH}(sk_{CH}, (h, r, m), m') \rightarrow r'$: Randomly choose $k \in Z_q^*$. Compute $R' = Q - kG$ and $s' = k - h'x$. Then $r' = (R', s')$.
5. $Verify_{CH}(h_{CH}, pk_{CH}, m, r) \rightarrow \{0, 1\}$: Compute $h = H(m)$ and $h'_{CH} = R + hY + sG$. The algorithm outputs 1 if $h_{CH} = h'_{CH}$, and otherwise outputs 0.

In order to analyze the security of the above chameleon hashing algorithm, we use the following security theorem:

Theorem 1. *The proposed chameleon hash is secure and key-exposure-free.*

From the Definition 1, we know that a chameleon hash is secure if it satisfies correctness and collision resistance. Key exposure free means that an adversary cannot calculate the private key even if he has a pair of collisions. It is clear that the chameleon hash is correct. The proofs of collision resistance and key exposure free can be referred to in [38,39], which are similar to ours.

3.2. Construction of CHRT

The general construction of the chameleon hash with revocable trapdoor (CHRT) is given in this section:

1. $\boldsymbol{Setup_{CHRT}(1^{\lambda}) \rightarrow (pp_1, pp_2)}$: Run the algorithm $Setup_{CH_1}(1^{\lambda})$ and $Setup_{CH_2}(1^{\lambda})$ to obtain the public parameter pp_1 and pp_2, and then return (pp_1, pp_2).
2. $\boldsymbol{KeyGen_{CHRT}(pp_1) \rightarrow (pk_{CH}, sk_{CH})}$: Run the algorithm $KeyGen_{CH_1}(pp_1)$ to obtain the chameleon key pair (pk_{CH}, sk_{CH}), and then return (pk_{CH}, sk_{CH}).
3. $\boldsymbol{KeyGen_{CHRT}(pp_2) \rightarrow (spk_{CH}, ssk_{CH})}$: Run the algorithm $KeyGen_{CH_2}(pp_2)$ to obtain the chameleon key pair (spk_{CH}, ssk_{CH}), and then return (spk_{CH}, ssk_{CH}).
4. $\boldsymbol{Hash_{CHRT}(pk_{CH}, spk_{CH}, m) \rightarrow (h, r)}$: Run the algorithm $Hash_{CH_1}(pk_{CH}, m)$ to obtain hash/check string pair (h_1, r_1). Then, run the algorithm $Hash_{CH_2}(spk_{CH}, h_1)$ to obtain hash/check string pair (h_2, r_2). Finally, return $(h, r) = (h_2, (h_1, r_1, r_2))$.
5. $\boldsymbol{SkAdapt_{CHRT}(sk_{CH}, m, m', r) \rightarrow r'}$: Phrase r as (h_1, r_1, r_2) and run the algorithm $Adapt_{CH_1}(sk_{CH}, (h_1, r_1, m), m')$ to obtain new check string r'_1. Then, return $r' = (h_1, r'_1, r_2)$.
6. $\boldsymbol{SskAdapt_{CHRT}(ssk_{CH}, m, m', h, r) \rightarrow r'}$: Phrase r as (h_1, r_1, r_2) and run the algorithm $Hash_{CH_2}(pk_{CH}, m', r_1)$ to obtain a new hash value h'_1. Then, run the algorithm $Adapt_{CH_2}(ssk_{CH}, (h_2, r_2, h_1), h'_1)$ to obtain a new check string r'_2. Finally, return $r' = (h'_1, r_1, r'_2)$.
7. $\boldsymbol{SkRevoke_{CHRT}(ssk_{CH}, spk_{CH}, m, h, r) \rightarrow (pk^*_{CH}, r^*)}$: Choose a new chameleon public key pk^*_{CH} using the algorithm $KeyGen_{CHRT}(pp^*_1)$. Then, run the algorithm $Hash_{CH_1}(pk^*_{CH}, m)$ to obtain r^*_1 and h^*_1. Run the algorithm $Adapt_{CH_2}(ssk_{CH}, (h_2, r_2, h_1), h^*_1)$ to obtain r^*_2. Finally, return $(pk^*_{CH}, r^*) = (pk^*_{CH}, (h^*_1, r^*_1, r^*_2))$.
8. $\boldsymbol{Verify_{CHRT}(pk_{CH}, spk_{CH}, m, h, r) \rightarrow \{0, 1\}}$: Phrase r as (h_1, r_1, r_2) and run the algorithm $Hash_{CH_1}(pk_{CH}, m, r_1)$ to obtain hash value h'_1. Then run $Hash_{CH_2}(spk_{CH}, h'_1, r_2)$ to obtain hash value h'_2. If $(h_2 = h'_2) \wedge (h_1 = h'_1)$, return 1; otherwise, return 0.

As shown above, CHRT consists of two chameleon hash functions which are represented by CH_1 and CH_2, respectively, where CH_2 uses the output of CH_1 as its input. By using such a structure, everyone can compute the chameleon hash by the algorithm $Hash_{CHRT}$ using pk_{CH} and spk_{CH} after executing algorithm $Setup_{CHRT}$ and $KeyGen_{CHRT}$. In addition, both sk_{CH} and ssk_{CH} holders can compute collisions by the algorithm $SkAdapt_{CHRT}$ and $SskAdapt_{CHRT}$, but the holder of ssk_{CH} can revoke the ability of the holder of sk_{CH} to compute collisions by the algorithm $SkRevoke_{CHRT}$. We can use this property to design permission revocation algorithms in data sharing.

4. The Proposed System

The proposed system and the operations involved in the system are introduced in detail in this section.

4.1. The System Model

We give the architecture of our system in Figure 2, which illustrates the operations that can be performed by the participants in the system. Specifically, there are five types of participants in our system:

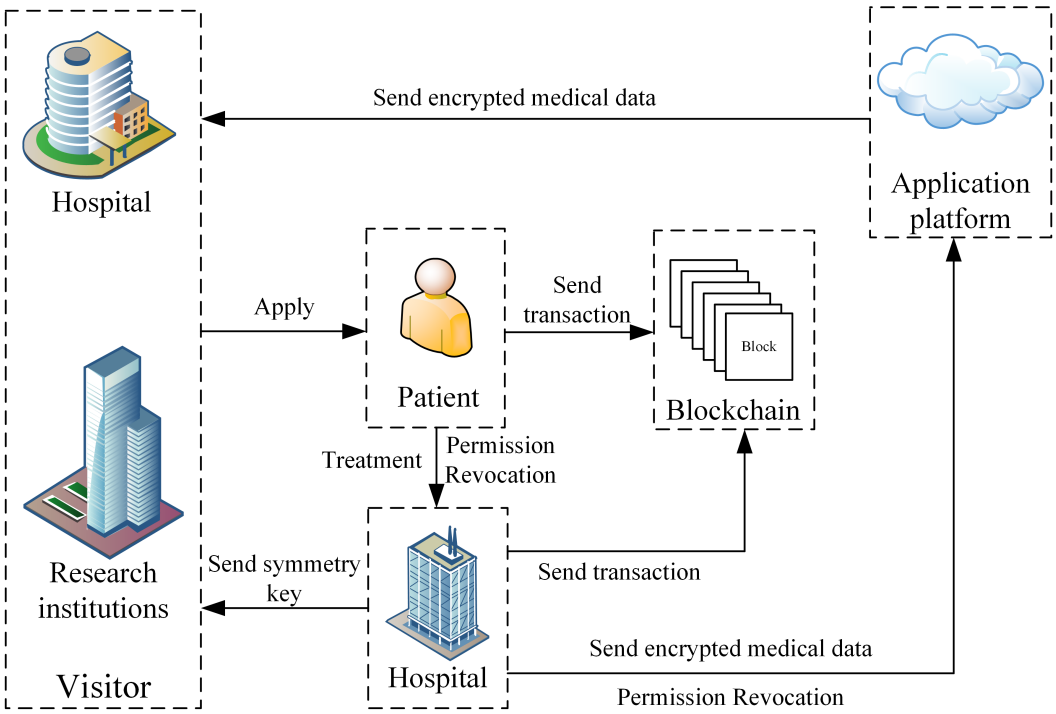


Figure 2. Architecture of our system.

Hospital: Hospital collects patients' medical data and submits their digests to the blockchain. After the patient is discharged, the hospital encrypts and submits the medical data to the application platform. When a visitor obtains permission from the patient, he needs to send the corresponding authorization information to the hospital. The hospital then sends the decryption key to him through a secure channel.

Patient: Medical data are generated during the patient's medical treatment in the hospital. Additionally, the patient can provide the visitor with permission to access the data.

Visitor: Visitor applies to the patient for permission to access the patient's medical data.

Blockchain: The blockchain stores transactions that include the identity of patients, digests of medical data and so on. Additionally, hospitals and patients will also submit verification transactions to the blockchain, which are used to verify the permission of the participant.

Application platform: The platform stores encrypted medical data sent by the hospital and the transactions published on the blockchain. When the platform receives a visitor's application, it signs the data and sends the signature along with the encrypted data to the visitor. The research [40] suggests that the application platform can provide essential support for research and learning.

In addition, the steps in Figure 3 are as follows:

1. When a hospital receives a new patient, the hospital must register him to grant him control over his own medical data.
2. During the visit of the patient, the hospital stores the generated medical data in the local database and submits the description and digest of the data to the blockchain.
3. When the patient is discharged, the hospital sends the encrypted medical data, description and digest of the entire medical data to the application platform.
4. The visitor applies for the patient to access his medical data. If the patient agrees to the visitor's request, he needs to sends access parameters to the visitor.
5. The visitor sends the access parameters to the hospital. If the access parameters is valid, the hospital sends the decryption key to the visitor.

6. The platform signs the medical data and sends them along with the encrypted data to the visitor.

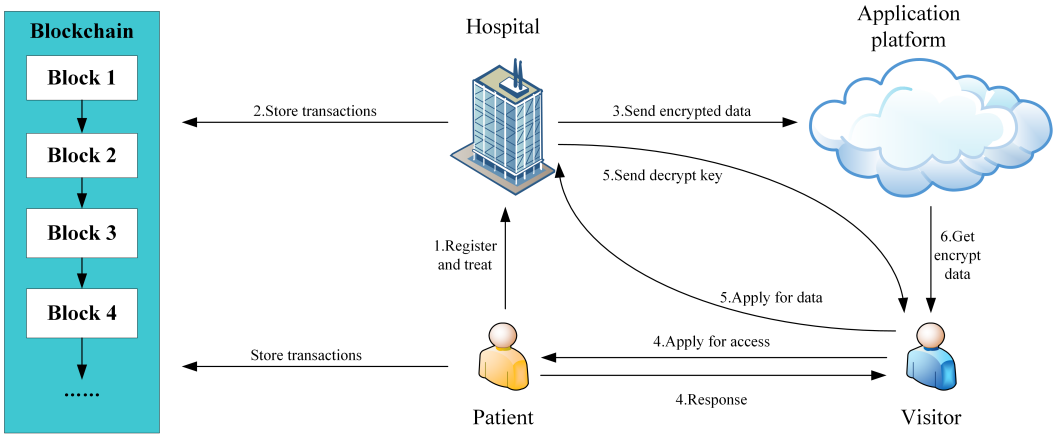


Figure 3. Workflow diagram of our system.

There are four operations in our medical data-sharing system: new patient registration, data storage, data sharing, and permission revocation. These operations are introduced in detail in the rest of this section, and the notations are described in Table 2.

Table 2. Notations.

Notation	Descriptions
$name$	Patient's identification information
(pk_{CH}, sk_{CH})	Hospital's chameleon key pair
(spk_{CH}, ssk_{CH})	Patient's chameleon key pair
(pk, sk)	Hospital's signature key pair
(pck, psk)	Visitor's chameleon key pair
(cpk, csk)	The validation public key and signature private key of the application platform
$H()$	The hash algorithm
$sign(), verify()$	The digital signature algorithm and the verification algorithm of digital signature
$encode^{sym}(), decode^{sym}()$	Symmetric encryption algorithm and symmetric decryption algorithm, which are used to encrypt and decrypt medical data
k_{id}	The symmetric key for encryption and decryption

4.2. New Patient Registration

When a hospital admits a new patient, the hospital must complete the registration process, which involves the following steps:

1. Hospital uses its chameleon public key pk_{CH} and patient's chameleon public key spk_{CH} to calculate

$$(h, r) = hash_{CHRT}(pk_{CH}, spk_{CH}, name) \quad (1)$$

2. The hospital selects a random number d and computes a new randomness r' using its chameleon private key sk_{CH}:

$$(h_1, r_1', r_2) = SkAdapt_{CHRT}(sk_{CH}, name, name||d, r) \quad (2)$$

3. The hospital obtains the chameleon public key of the application platform. Then, it calculates $cert = sign_{sk}(cpk||r_1')$ using its private key sk and then sends $(name, r_1{}', cert)$

to the application platform to give it proxy signing authority, where *sign* is the digital signature algorithm.

4. The hospital sets $version = 1$ and computes

$$\sigma_{tran} = sign_{sk}(verify, name, version, d, r, pk_{CH}, spk_{CH}) \quad (3)$$

where $verify$ represents the kind of transaction.

5. The hospital submits the following transaction to the blockchain:

$$tran_v = (ver, name, version, d, r, pk_{CH}, spk_{CH}, (\sigma_{tran}, pk)) \quad (4)$$

4.3. Data Storage

As shown in Figure 4, medical data are generated in the system one by one. In the process of a patient's medical treatment, his medical data cannot be generated at the same time but at different time or places. If the medical data record is uploaded to the blockchain when the patient is discharged, there is a risk of tampering. Therefore, we decide to submit the digest of the medical data to the blockchain when they are just generated.

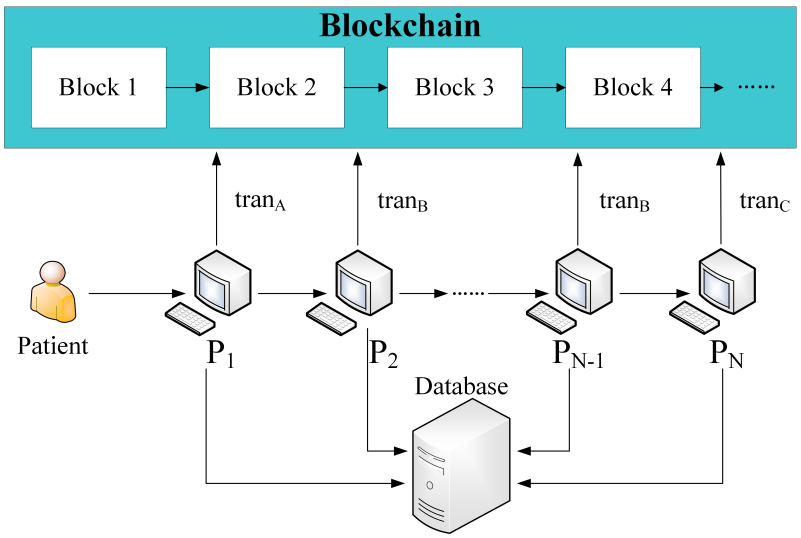


Figure 4. Generation of medical data.

The transactions submitted to the blockchain are divided into three categories—*A*, *B*, and *C*—and use DG_i to represent the hospital department that generates the corresponding medical data. Next, we will describe in detail how to deal with these three kinds of data separately in our algorithm.

When a patient seeks medical treatment, his first piece of medical data m_1, which is generated by DG_1, is usually the admission information. DG_1 needs to use the following algorithm to store this type of data:

1. DG_1 sets $num_1 = 1$ and computes $\sigma_1 = sign_{sk_1}(m_1, id, num_1)$ and $h_{\sigma_1} = H(\sigma_1)$, where sk_k is the signature key of DG_k and id is the identifier of a series of medical data generated by this medical treatment.
2. DG_1 computes $\sigma_A = sign_{sk_1}(id, A, addr_1, num_1, h_{\sigma_1})$, where A represents this kind of medical data, and $addr_1$ contains brief information about the piece of medical data, such as the description and the time when the data were generated.
3. DG_1 submits $tran_A = (id, A, addr_1, num_1, h_{\sigma_1}, (\sigma_A, pk_1))$ to the blockchain, which will return the hash value $tran_{hash_1}$ of the transaction to DG_1.
4. DG_1 saves $(M_1, tran_{hash_1})$ in the local database, where $M_1 = ((m_1, id, num_1), (\sigma_1, pk_1))$.

The second type of medical data is generated during the treatment by DG_i and denoted by $m_i (1 < i < N)$. For this type of data, DG_i performs the following operations:

1. DG_i obtains $tran_{hash_{i-1}}$ by id in the hospital's local database.
2. DG_i sets $num_i = i$ and computes $\sigma_i = sign_{sk_i}(m_i, id, num_i)$ and $h_{\sigma_i} = H(\sigma_i)$.
3. DG_i computes $\sigma_{B_i} = sign_{sk_i}(id, B, addr_i, num_i, h_{\sigma_i}, tran_{hash_{i-1}})$, where B represents this of medical data.
4. DG_i submits the transaction $tran_{B_i} = (id, B, addr_i, num_i, tran_{hash_{i-1}}, h_{\sigma_i}, (\sigma_{B_i}, pk_i))$ to the blockchain and obtains the hash value $tran_{hash_i}$ of this transaction.
5. DG_i saves $(M_i, tran_{hash_i})$ in the local database, where $M_i = ((m_i, id, num_i), (\sigma_i, pk_i))$.

The third type of medical data is generated when a patient is discharged from the hospital after completing the medical procedure. The last medical data m_N is generated by DG_N as follows:

1. DG_N obtains $tran_{hash_{N-1}}$ by id in the hospital's local database.
2. DG_N sets $num_N = N$ and computes $\sigma_N = sign_{sk_N}(m_N, id, num_N)$, $h_{\sigma_N} = H(\sigma_N)$ and $h = H(pk||m^N)$, where $m^N = (m_i, id, num_i)_{i=1}^N$.
3. DG_N generates a symmetry key k_{id} and saves M_N and k_{id} in the local database.
4. DG_N computes $\sigma_C = id, C, num_N, addr_N, tran_{hash_{N-1}}, h_{\sigma_N}, h)$, where C represents the kind of transaction.
5. DG_N submits the transaction $tran_C = (id, C, num_N, tran_{hash_{N-1}}, h_{\sigma_N}, h, (\sigma_C, pk_N))$ to the blockchain.
6. DG_N computes $C_m = encode^{sym}(k_{id}, m^N)$ and sends (id, C_m, h) to the application platform. In our article, we choose the AES algorithm for encrypting and decrypting medical data.

Through the above algorithm, we submit the digest of the patient's medical data to the blockchain and the ciphertext of the complete medical data to the application platform. We also store the hash value of the previous transaction in the current transaction, allowing visitors to easily search for a complete transaction chain of the patient.

4.4. Data Sharing

In this operation, the visitor first needs to request the patient to access the medical data. If the patient agrees on the application, he performs the corresponding calculation and sends the result to the visitor. The visitor then applies to the hospital through the application platform for the data. The hospital verifies the eligibility of the visitor. After the verification is passed, the hospital sends the symmetric key k_{id} to the visitor. After obtaining the encrypted data and signature from the application platform, the visitor only needs to perform the decryption operation to obtain the medical data and verify their validity.

4.4.1. Permission Acquisition

The visitor sends a request to the patient to access his data marked with id. If the patient agrees to the request, he needs to calculate the following:

$$r' = SskAdapt_{CHRT}(ssk_{CH}, pk_{CH}, name||id||pck, h, r) \tag{5}$$

and sends r' to the visitor, where pck is the visitor's chameleon public key.

After the visitor receives r', he sends (id, r') to the hospital for verification through a secure channel. Then the hospital calculates the following:

$$b = Verify_{CHRT}(pk_{CH}, spk_{CH}, name||id||pck, h, r') \tag{6}$$

If $b = 1$, it proves that the visitor has the permission to obtain the data. The hospital then needs to send the symmetric key k_{id} of the data requested by the visitor to him through a secure channel.

According to the collision-resistant characteristics of the chameleon hash, for a party who does not know the chameleon private key, even if he owns m, r and m', he cannot

calculate r' such that $Hash_{CH}(r, m) = Hash_{CH}(r', m')$. If the visitor can pass the verification, then the hospital can fully trust that the visitor has obtained the permission.

4.4.2. Data Acquisition

The application platform finds (h, C_m) by id and run the algorithm $Hash_{CH}$ to obtain (r_{CH}, h_{CH}). To be specific, it randomly selects $r_{CH} = (R, s)$ and computes $h_{CH} = R + hY + sG$. Then the platform computes $\sigma = sign_{csk}(h_{CH})$. Through the above calculation, the chameleon signature $\sigma_{CH} = (r_{CH}, r_1', cpk, cert, \sigma)$ is calculated. Finally, the application platform sends C_m and σ_{CH} to the visitor. Visitor can decrypt the ciphertext to obtain the data by $m^N = decode^{sym}(C_m)$, and confirm the validity of the signature through the following steps.

Firstly, the visitor needs to confirm the validity of $cert$ and σ. If $verify(pk, cpk||r_1', cert) = 1$, then $cert$ is valid, where $verify$ is the verification algorithm of the digital signature. If $verify(cpk, pk||m^N, \sigma) = 1$, it proves that σ is valid.

Next, the visitor needs to confirm whether r_1' is valid. He should obtain the verification transaction $tran_v = (ver, name, version, d, r, pk_{CH}, spk_{CH})$ from the blockchain first, and then calculate

$$b = Verify_{CHRT}(pk_{CH}, spk_{CH}, name||d, h, (h_1, r_1', r_2)) \tag{7}$$

If $b = 1$, then it proves that r_1' is valid, which means that the application platform is eligible to sign.

In such a case, only the visitor can verify the validity of the signature. Next, we will describe how we use the chameleon hash function to protect the privacy of the patient.

If the visitor wants to disclose the message m to an unauthorized third party, they must send both σ_{CH} and m to the party. If the visitor sends the signature σ generated by itself or only sends the medical data m to the third party, he will not be trusted by the third party.

However, even if the third party receives (σ_{CH}, m), he still cannot confirm whether the medical data m is the original data or has been tampered with by the visitor. This is because if the visitor changes m to m', he can calculate a collision using its trapdoor psk, which keeps the chameleon hash value h'_{CH} of m' equal to the original value h_{CH}. As a result, the chameleon signature remains unchanged. Since the third party knows that the visitor has this ability, they cannot determine whether the message has been tampered with, and therefore they can only choose not to trust the medical data. This operation ensures that medical data will not be leaked to any unauthorized parties.

4.5. Permission Revocation

The permission revocation algorithm is divided into two parts: the hospital revokes the application platform's permission, and the patient revokes the hospital's permission.

Firstly, the hospital gives the signing right to the application platform, allowing it to sign the medical data instead of the hospital. However, the application platform may not respond to legitimate signature requests or collude with a visitor who has obtained medical data to share the data with an unauthorized party.

Secondly, the hospital itself may exhibit malicious behaviors. It may not respond to the requests from verified visitors, or share medical data without consent, which could severely compromise patient privacy.

Therefore, the following two algorithms are designed to revoke the corresponding permissions.

4.5.1. Application Permission Revocation

The hospital randomly selects a number d' and a new chameleon public key cpk'. Then it computes $r_1'' = Adapt_{CH}(sk_{CH}, (h_2, r_1, name), name||d')$ and $cert' = sign(sk, cpk'||r_1'')$,

and sets $version = version + 1$, where $cert'$ is used to transfer the permission to another application platform. Finally, the hospital submits the following transaction to the blockchain:

$$tran_v' = (ver, name, version, d', r_2, pk_{CH}, spk_{CH}) \tag{8}$$

After the permission is revoked, if the visitor wants to verify whether r_1' is valid, he first gets the verification transaction $tran_v'$ from the blockchain and then calculates

$$h_1' = Hash_{CH}(r_1', name||d', pk_{CH}) \tag{9}$$
$$h_2' = Hash_{CH}(r_2, h_1', spk_{CH}) \tag{10}$$

It is obvious that $h_2' \neq h_2$, which proves that r_1' is invalid.

The public key can be selected randomly, or can be the chameleon public key of another application platform. After the revocation operation, even though the original application platform cannot pass the verification, the party corresponding to the new public key can pass the verification, which makes it possible to transfer permissions. The process of passing the verification with the new public key is as follows. The validator obtains the transaction $tran_v' = (verify, name, version, d', r_2, pk_{CH}, spk_{CH})$ from the blockchain, then calculates

$$h_1'' = Hash_{CH}(r_1'', name||d', pk_{CH}) \tag{11}$$
$$h_2'' = Hash_{CH}(r_2, h_1'', spk_{CH}) \tag{12}$$

Since $r_1'' = Adapt_{CH}(sk_{CH}, (h_2, r_1, name), name||d')$ there is $h_2'' = h_2$, so r_1'' is valid.

Through this algorithm, the hospital gains the ability to revoke the permission of the application platform, or transfer the permission to another application platform, which significantly improves the scalability of the system and provides more directions for its future application. Please note that in our article, scalability refers to the system's ability to be further developed.

4.5.2. Hospital Permission Revocation

In order to revoke the hospital's permission to manage the data, the patient needs to set $version = version + 1$ and compute

$$(pk^*, r^*) = SkRevoke_{CHRT}(ssk_{CH}, spk_{CH}, name, h, r) \tag{13}$$

Then, he submits the following transaction to the blockchain:

$$tran_v^* = (ver, name, version, d, r^*, pk_{CH}^*, spk_{CH}) \tag{14}$$

Since h_1^* is calculated by the patient using the new public key pk^*, the hospital no longer has the ability to calculate the collision of h_1^*, which means that the hospital loses the permission to manage the data.

5. Analysis of the System

5.1. The Security of CHRT

As mentioned in our contribution, we propose a chameleon hash with revocable trapdoor (CHRT) for distributing and validating permissions. Similar to [36], we give the definition about the security of CHRT as follows to analyze the security of CHRT.

Definition 2 (Secure CHRT)**.** *A secure chameleon hash with revocable trapdoor is secure if it satisfies correctness, collision resistance, revocability and permission revocation resistance.*

Correctness: Correctness means that all the hash/check string pair (h, r) correctly generated from $Hash_{CHRT}$, $SkAdapt_{CHRT}$, $SskAdapt_{CHRT}$ or $SkRevoke_{CHRT}$ can pass the verification of $Verify_{CHRT}$. According to the [36], it is clear that CHRT is correct.

Collision resistance: An adversary cannot calculate the collision of a certain hash value without knowing the private key sk_{CH} or ssk_{CH}.

Revocability: The owner of ssk_{CH} can make the collision calculated by the owner of sk_{CH} unable to pass the verification through the algorithm and the way of publishing some revocation parameters.

Permission revocation resistance: The party that does not own ssk_{CH} cannot revoke the permission that the collision calculated by the sk_{CH} owner can be verified.

Theorem 2. *If the underlying chameleon hash is secure, then CHRT is also secure.*

Lemma 1. *If the chameleon hash that makes up CHRT is secure, then the CHRT is collision-resistant.*

Proof. In CHRT, after calculating the chameleon hash value h of a message m using the algorithm $Hash_{CHRT}$, if a modifier wants to change the message m to m' without changing h, he needs to execute the algorithms $SkAdapt_{CHRT}$ or $SkAdapt_{CHRT}$. For the owner of sk_{CH}, he needs to run algorithm $Adapt_{CH_1}(sk_{CH}, (h_1, r_1, m), m')$ to obtain a new randomness r'. For the owner of ssk_{CH}, he first needs to run algorithm $Hash_{CH_2}(pk_{CH}, m', r_1)$ to obtain a new hash value h'_1 and then run algorithm $Adapt_{CH_2}(ssk_{CH}, (h_2, r_2, h_1), h'_1)$ to obtain a new randomness r'. If an adversary $\mathcal{A}$ without sk_{CH} and ssk_{CH} wants to break the collision resistance of $CHRT$, he needs to break the collision resistance of CH_1 or CH_2. From this, it can be inferred that the probability of $\mathcal{A}$ breaking the collision resistance of $CHRT$ is $p = p_1 + p_2$, where p_1 and p_2 are the probability that $\mathcal{A}$ can break the collision resistance of the chameleon hash that makes up CHRT. Since p_1 and p_2 are negligible, p is negligible. □

Lemma 2. *If the chameleon hash that makes up CHRT is secure, then the CHRT is revocability.*

Proof. In our $SkRevoke_{CHRT}$ algorithm, we regenerate the key pair (pk', sk') and publish (pk', r') on the blockchain after revocation. Therefore, the probability that the revocability of our $SkRevoke_{CHRT}$ algorithm is broken is equal to the probability that the regenerated key pair (pk', sk') is exactly the same as the original key pair (pk, sk), which is negligible. □

Lemma 3. *The CHRT is permission-revocation-resistant if the chameleon hash that makes up CHRT is secure.*

Proof. In CHRT, after calculating the chameleon hash value h of a message m using the algorithm $Hash_{CHRT}(pk_{CH}, spk_{CH}, m)$, if the owner of ssk_{CH} wants to revoke the ability of the owner of sk_{CH} to calculate collisions, he needs to execute the algorithm $SkRevoke_{CHRT}$. Specifically, he first needs to execute the algorithm $KeyGen_{CHRT}(pp_1^*)$ to obtain a new chameleon key pair (pk_{CH}^*, sk_{CH}^*) and then run the algorithms $Hash_{CH_1}$ and $Adapt_{CH_2}$ to obtain the check string r^*. If an adversary $\mathcal{A}$ without ssk_{CH} wants to break the permission revocation resistance of $CHRT$, he needs to break the collision resistance of CH_2. From this, it can be inferred that the probability of $\mathcal{A}$ breaking the permission revocation resistance of $CHRT$ is p, where p is the probability that $\mathcal{A}$ can break the collision resistance of CH_2. According to Theorem 3, since the chameleon hash that makes up $CHRT$ is secure, p is negligible. □

5.2. Data Access Control

Our system relies on cryptographic security to ensure access control. If visitor v wants to access medical data m of patient p, he needs to apply to the patient. If the patient agrees to the visitor's request, he computes the hash value h'_1 of $name||id||pck$, and then computes the collision r'_2 of h'_1 and h_1 using ssk_{CH}. Visitor v then sends r'_2 to the hospital for verification. If r'_2 is valid, the hospital sends the decryption key to visitor v. Since the visitor cannot forge r'_2 without ssk_{CH}, it is impossible for v to obtain the data without authorization.

5.3. Data Security

The validity of the chameleon signature can only be verified by the recipient designated by the signer, and no one else can confirm whether the signature was generated by the signer or forged by the designated recipient. That is, the recipient cannot convince any third party by transferring the signer's signature. Using the non-transferable chameleon signature can ensure that the data will not be leaked by the recipient after being shared, thus protecting the rights and interests of the data owner.

Theorem 3 (Non-Transferability)**.** *A chameleon signature is non-transferable if the private key of chameleon hash is not leaked.*

Proof. Suppose that an adversary $\mathcal{A}$ can share the acquired data with a third party $\mathcal{C}$ without permission. We prove that $\mathcal{C}$ cannot tell whether the data leaked by $\mathcal{A}$ was tampered with by $\mathcal{A}$.

The adversary $\mathcal{A}$ sends medical data m and signature $\sigma_{CH} = (r_{CH}, r_1^{'}, cpk, cert, \sigma)$ signed by application platform to $\mathcal{C}$. $\mathcal{C}$ needs to compute $verify(pk, cpk||r_1^{'}, cert)$ to verify the validity of the $cert$ and compute $verify(cpk, pk||m^N, \sigma)$ to verify whether m^N is valid.

However, $\mathcal{A}$ can also generate such a signature by following the steps below. First, $\mathcal{A}$ obtains the original signature $\sigma_{CH} = (r_{CH}, r_1^{'}, cpk, cert, \sigma)$. Then he chooses a new message m', and computes $r'_{CH} \leftarrow Adapt_{CH}(sk_{CH}, (h, r_{CH}, m), m')$, where sk_{CH} is $\mathcal{A}$'s chameleon private key. $\mathcal{A}$ then replaces r_{CH} in signature σ_{CH} with r'_{CH}. Finally, $\mathcal{A}$ sends the replaced signature σ'_{CH} and the new message m' to $\mathcal{C}$.

When $\mathcal{C}$ verifies the validity of $\sigma'_{CH} = (r', r_1^{'}, cpk, cert, \sigma)$, it can still pass the verification. The validity verification step of $cert$ is $verify(pk, cpk||r_1^{'}, cert)$, and if it is valid, $\mathcal{C}$ continues to verify the validity of the signature σ through $verify(cpk, pk||m', \sigma)$. Although the message changes from m to m', since $\mathcal{A}$ replaces r_{CH} with r'_{CH}, it can still pass the verification. Since $\mathcal{A}$ has already calculated the collision of m and m', there is $h'_{CH} = R' + h'Y + s'G = h_{CH}$, that is, $\sigma' = sign(sk, h'_{CH}) = \sigma$, which means that the signature can still be verified.

After $\mathcal{C}$ receives (m^*, σ^*_{CH}) sent by $\mathcal{A}$, since $\mathcal{A}$ owns the chameleon private key, $\mathcal{C}$ cannot tell whether (m^*, σ^*_{CH}) are the original signature and message, or forged by $\mathcal{A}$. Therefore, $\mathcal{C}$ can only think that the data are invalid. This ensures that shared medical data are not shared with unauthorized parties. □

6. Experiments

To examine the efficiency of our system, we performed experiments in several aspects and compared ours with existing systems [26,27,35,36]. We compare the time for computing chameleon hash, permission distribution, data encryption and decryption in our system and existing systems. The data storage overhead is also compared. Additionally, we measure the time required for permission revocation in our system. Since the computation of data in system [28] differs depending on the usage of the data, we do not compare our system with it. The experiments are based on the Ethereum blockchain and are conducted on a computer with 16 GB RAM and AMD Ryzen 5 4600H CPU@3.00 GHz running Win 10, and JPBC 2.0.0 library is used to generate the elliptic curve group. In addition, unlike machine learning, the focus of our system is on protecting the privacy of data. Therefore, as shown in Figure 5, we followed the process of patients seeking medical care and recorded the condition of the human body at different times and places multiple times, such as body temperature and blood pressure. The above information was packaged as medical data to conduct experiments. As our system does not involve statistics, we did not use the methods provided in [41].

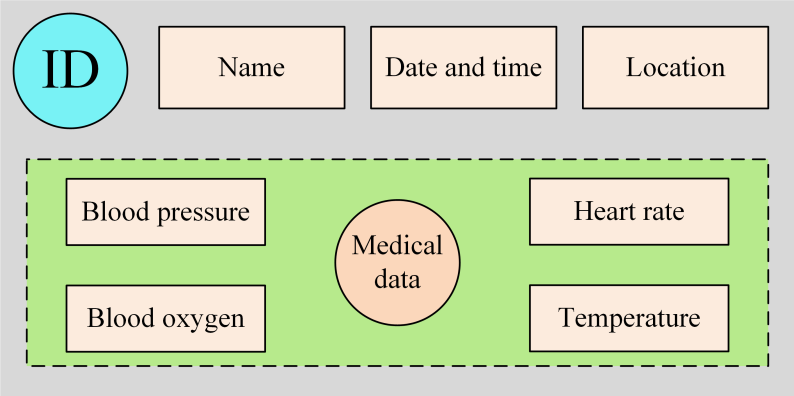


Figure 5. The sample of medical data.

6.1. Computational Costs of Chameleon Hash

The chameleon hash based on the elliptic curve group is proposed in Section 3.1, and its computational cost is analyzed in this section. We implemented the chameleon hashes as described in our scheme and the scheme of [35,36] on the same computer and software. As shown in Figure 6, we compare the computational cost of our scheme and schemes of [35,36]. In [35], a discrete logarithm was used to build a chameleon hash, while the latter proposes an RSA-based chameleon hash. We test the time token for computing the chameleon hash with different lengths of message m. It can be seen that under the condition that the length of message m exceeds 3000, the time cost in our scheme is the lowest. Therefore, our scheme has higher efficiency in computing the chameleon hash.

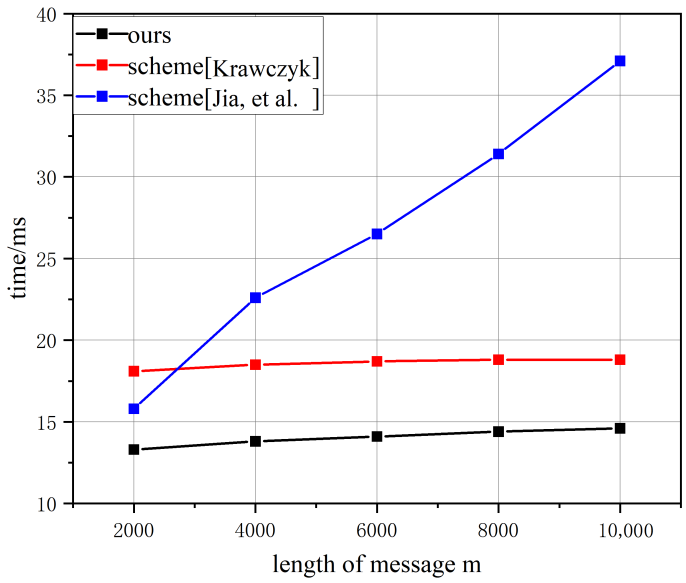


Figure 6. Time consumption of computing chameleon hash [35,36].

6.2. Computational Costs of Permission Operation

The time taken for the data owner to approve a visitor's request is tested in this section. As shown in Figure 7, the time overhead of permission acquisition in [26] is proportional to the number of attributes. However, in our system, the computational cost of permission acquisition is almost constant. This is because, unlike the system of [26] that uses the key generation algorithm in attribute-based encryption to distribute permissions, the permission acquisition in our system only needs to calculate a collision of the chameleon hash.

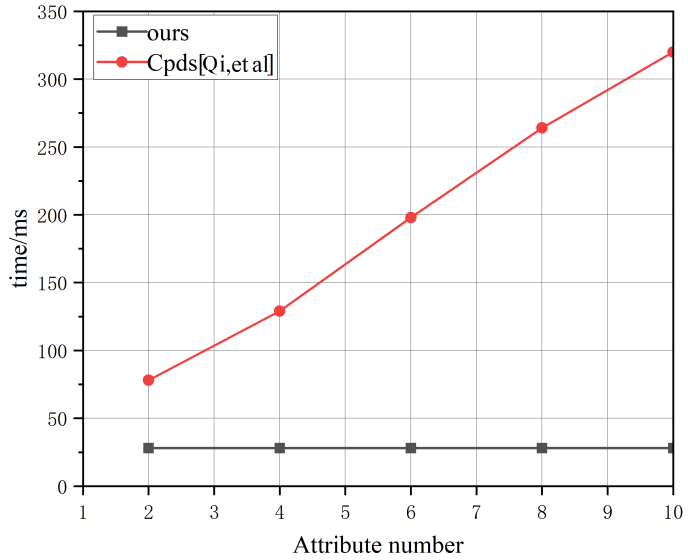


Figure 7. Time consumption of permission acquisition [26].

Next, we test the time overhead of permission revocation in our system. The results are shown in Figure 8. Multiple experiments are conducted to measure the time consumption for different times of revoking permissions. It can be seen that the time spent on revocation increases with the number of revocations, and the time for a single revocation does not exceed 30 ms. Therefore, we think that this time overhead is acceptable in our system.

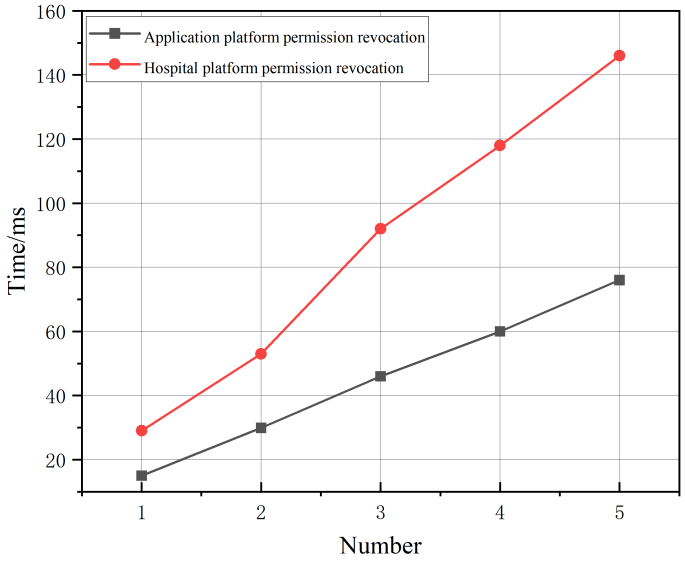
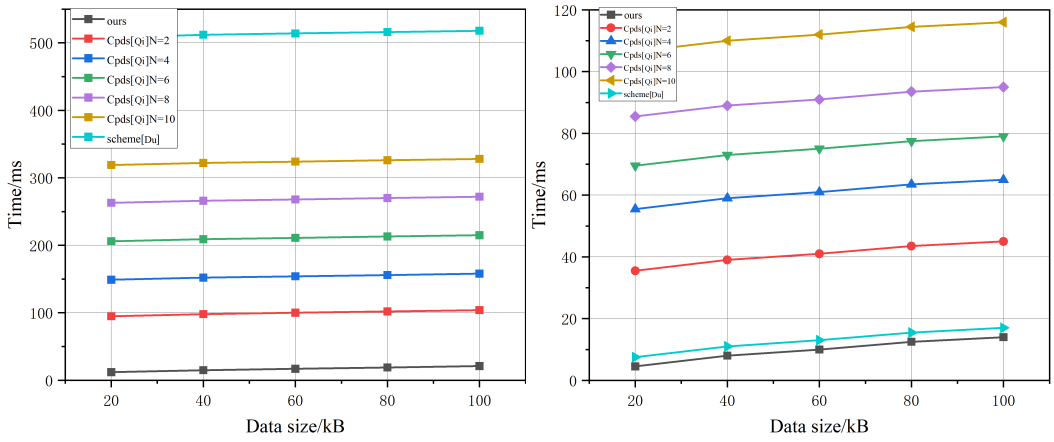


Figure 8. Time consumption of permission revocation.

6.3. Computational Costs of Encryption and Decryption

We conduct experiments on the existing system of [26,27] and ours to compare the time overhead of data encryption and decryption. The time consumption of data encryption is shown in Figure 9a. It can be seen that the encryption time consumption of data in the system [26] is proportional to the number of attributes and the size of the data. In our system, the encryption time of the data is only related to the size of them. In the system of [26,27], the data are first encrypted using symmetric encryption, and then the symmetric encryption key is encrypted using attribute-based encryption and ElGamal encryption, respectively. In contrast, only symmetric encryption is required in our system, making it more efficient than the existing systems in terms of processing.

Figure 9b shows the decryption time of the system of [26,27]. In the system of [26], the decryption is divided into two stages: symmetric decryption and attribute-based decryption. In the system of [27], symmetric decryption and ElGamal decryption are required. It can be seen from the figure that the decryption time of the system of [26] is proportional to the number of attributes and the size of the data, while it is only related to the size of data in the system of [27]. In our system, only symmetric decryption is required, resulting in higher efficiency compared to the existing systems.



(**a**) Encrypt data (**b**) Decrypt data

Figure 9. Time consumption of encrypt and decrypt data [26,27].

6.4. Data Storage Overhead

If the original data are compressed and stored on the blockchain according to the algorithm of the system of [26], the storage overhead of the data-sharing system is greatly increased. Therefore, instead of storing the complete data, we decide to just store important information about them on the blockchain. Since the data need to be processed by a specific algorithm before being submitted to the blockchain, we first measure the size of the processed data in Cpds [26] and our system, respectively, when the size of the original data is different. Then we calculate the ratio of the size of the original data to the size of the processed data in Cpds [26] and our system, respectively, shown in Table 3. The larger the ratio, the smaller the data storage overhead. As shown in Table 3, the data storage overhead in our system is smaller than that of Cpds [26].

Table 3. Data storage overhead.

System	100 B	1 kB	5 kB	10 kB
Cpds [26]	1.23	1.40	1.44	1.45
ours	1.56	15.63	78.12	156.25

7. Conclusions

Since the existing systems generally lack the ability to restrict the power of participants and prevent visitors from sharing obtained data to unauthorized parties, this article proposed a blockchain-based medical data-sharing system to tackle these problems. Firstly, medical data are encrypted by symmetric encryption and stored on application platforms to protect the privacy of the medical data. Secondly, patients have the ability to authorize visitors to access their medical data in this system by using a chameleon hash. Thirdly, the power of hospitals and the application platform is limited and can be revoked in the system by our proposed CHRT. Finally, our system employs a chameleon signature to prevent visitors from sharing the obtained data with unauthorized parties. The security analysis and experimental results show that our system is secure and has excellent performance

compared to other systems. In future research, we will explore ways to more effectively restrict and transfer permissions.

Author Contributions: Conceptualization, M.H. and Y.R.; Software, M.H.; Validation, C.C.; Writing—original draft, M.H.; Writing—review & editing, Y.R.; Supervision, Y.R.; Funding acquisition, Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Natural Science Foundation of Shanghai (20ZR1419700, 22ZR1481000), and Henan Key Laboratory of Network Cryptography Technology (LNCT2021-A13).

Acknowledgments: The authors would like to thank the anonymous reviewers for their helpful comments.

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

References

1. Engelhardt, M. Hitching Healthcare to the Chain: An Introduction to Blockchain Technology in the Healthcare Sector. *Technol. Innov. Manag. Rev.* **2017**, *12*, 22–34. [CrossRef]
2. Pazaitis, A.; De Filippi, P.; Kostakis, V. Blockchain and Value Systems in the Sharing Economy: The Illustrative Case of Backfeed. *Technol. Forecast. Soc. Chang.* **2017**, *12*, 105–115. [CrossRef]
3. Fiore, M.; Capodici, A.; Rucci, P.; Bianconi, A.; Longo, G.; Ricci, M.; Sanmarchi, F.; Golinelli, D. Blockchain for the Healthcare Supply Chain: A Systematic Literature Review. *Appl. Sci.* **2023**, *13*, 686. [CrossRef]
4. Zhou, N.; Long, S.; Liu, H.; Liu, H. Structure—Attribute Social Network Graph Data Publishing Satisfying Differential Privacy. *Symmetry* **2022**, *14*, 2531. [CrossRef]
5. Colomo-Palacios, R.; Sánchez-Gordón, M.; Arias-Aranda, D. A critical review on blockchain assessment initiatives: A technology evolution viewpoint. *J. Softw. Evol. Process.* **2020**, 32, e2272. [CrossRef]
6. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 14 April 2021).
7. Yong, Y.; Wang, F. Blockchain: The State of the Art and Future Trends. *Acta Autom. Sin.* **2016**, *42*, 481–494.
8. Yeh, K.-H.; Yang, G.-Y.; Butpheng, C.; Lee, L.-F.; Liu, Y.-H. A Secure Interoperability Management Scheme for Cross-Blockchain Transactions. *Symmetry* **2022**, *14*, 2473. [CrossRef]
9. Wang, H.; Wang, Q.; He, D. Blockchain-Based Private Provable Data Possession. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2379–2389. [CrossRef]
10. Ma, Z.; Ma, J.; Miao, Y.; Li, Y.; Deng, R.H. ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1639–1654. [CrossRef]
11. Ma, Z.; Ma, J.; Miao, Y.; Liu, X.; Choo, K.-K.R.; Deng, R. Pocket diagnosis: Secure federated learning against poisoning attack in the cloud. *IEEE Trans. Serv. Comput.* **2022**, *15*, 3429–3442. [CrossRef]
12. Weng, J.; Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2438–2455. [CrossRef]
13. Swan, M. *Blockchain: Blueprint for a New Economy*; O'Reilly Media, Inc.: Newton, MA, USA, 2015.
14. Ahmad, M.; Khan; Salah, K. IoT Security: Review, Blockchain Solutions, and Open Challenges. *Future Gener. Comput. Syst.* **2018**, *82*, 395–411. [CrossRef]
15. Zhang, R.; Xue, R.; Liu, L. Security and Privacy for Healthcare Blockchains. *IEEE Trans. Serv. Comput.* **2022**, *15*, 3668–3686. [CrossRef]
16. Saeed, H.; Malik, H.; Bashir, U.; Ahmad, A.; Riaz, S.; Ilyas, M.; Bukhari, W.A.; Khan, M.I.A. Blockchain technology in healthcare: A systematic review. *PLoS ONE* **2022**, *17*, e0266462. [CrossRef]
17. Yin, H.L.; Fu, Y.; Li, C.L.; Weng, C.X.; Li, B.H.; Gu, J.; Lu, Y.S.; Huang, S.; Chen, Z.B. Experimental quantum secure network with digital signatures and encryption. *Natl. Sci. Rev.* **2022**, *10*, nwac228. [CrossRef]
18. Bennett, C.H.; Brassard, G. Quantum cryptography: Public-key distribution and coin tossing. In Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, 9–12 December 1984; pp. 175–179.
19. Lucamarini, M.; Yuan, Z.L.; Dynes, J.F. Overcoming the rate-distance limit of quantum key distribution without quantum repeaters. *Nature* **2018**, *557*, 400–403. [CrossRef]
20. Xie, Y.M.; Lu, Y.S.; Weng, C.X.; Cao, X.Y.; Jia, Z.Y.; Bao, Y.; Wang, Y.; Fu, Y.; Yin, H.L.; Chen, Z.B. Breaking the Rate-Loss Bound of Quantum Key Distribution with Asynchronous Two-Photon Interference. *PRX Quantum 3* **2022**, *3*, 020315. [CrossRef]
21. Gu, J.; Cao, X.Y.; Fu, Y.; He, Z.W.; Yin, Z.J.; Yin, H.L.; Chen, Z.B. Experimental measurement-device-independent type quantum key distribution with flawed and correlated sources. *Sci. Bull.* **2022**, *67*, 2167–2175. [CrossRef]
22. Kassab, M.; DeFranco, J.; Malas, T. Exploring Research in Blockchain for Healthcare and a Roadmap for the Future. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 1835–1852. [CrossRef]
23. Pan, H.; Zhang, Y.; Si, X.; Yao, Z.; Zhao, L. MDS^2-C^3PF: A Medical Data Sharing Scheme with Cloud-Chain Cooperation and Policy Fusion in IoT. *Symmetry* **2022**, *14*, 2479. [CrossRef]

24. Rahulamathavan, Y.; Phan, R.; Rajarajan, M. Privacy-Preserving Blockchain Based IoT Ecosystem using Attribute-based Encryption. In Proceedings of the 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems, Bhubaneswar, India, 17–20 December 2017; pp. 1–6.
25. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 20–23 May 2007; pp. 321–334.
26. Qi, S.; Lu, Y.; Zheng, Y.; Li, Y.; Chen, X. Cpds: Enabling Compressed and Private Data Sharing for Industrial Internet of Things over Blockchain. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2376–2387. [CrossRef]
27. Du, M.; Chen, Q.; Chen, J. An Optimized Consortium Blockchain for Medical Information Sharing. *IEEE Trans. Eng. Manag.* **2020**, *68*, 1677–1689. [CrossRef]
28. Wang, Y.; Su, Z.; Zhang, N. SPDS: A Secure and Auditable Private Data Sharing Scheme for Smart Grid Based on Blockchain and Smart Contract. *IEEE Trans. Ind. Inform.* **2020**, *17*, 7688–7699. [CrossRef]
29. Zhang, N.; Li, J.; Lou, W.; Hou, Y.T. PrivacyGuard: Enforcing Private Data Usage with Blockchain and Attested Execution. In *International Workshop on Data Privacy Management*; Springer: Berlin, Germany, 2018; pp. 345–353.
30. Nguyen, D.; Pathirana, P.; Ding, M. BEdgeHealth: A Decentralized Architecture for Edge-Based IoMT Networks Using Blockchain. *IEEE Internet Things J.* **2021**, *8*, 11743–11757. [CrossRef]
31. Wu, G.; Wang, S.; Ning, Z.; Zhu, B. Privacy-Preserved Electronic Medical Record Exchanging and Sharing: A Blockchain-Based Smart Healthcare System. *IEEE J. Biomed. Health Inform.* **2022**, *26*, 1917–1927. [CrossRef]
32. Yu, C.; Zhan, Y.; Sohail, M. SDSM: Secure Data Sharing for Multilevel Partnerships in IoT Based Supply Chain. *Symmetry* **2022**, *14*, 2656. [CrossRef]
33. Costan, V.; Devadas, S. Intel SGX Explained. Available online: https://eprint.iacr.org/2016/086.pdf (accessed on 17 September 2021).
34. Ren, Y.; Zhang, X.; Gu, D.; Feng, G. Efficient outsourced extraction of histogram features over encrypted images in cloud. *Sci. China Inf. Sci.* **2021**, *64*, 139105. [CrossRef]
35. Krawczyk, H.; Rabin, T. Chameleon Hashing and Signatures. In Proceedings of the 7th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 3–4 February 2000; pp.143–154.
36. Jia, Y.; Sun, S.; Zhang, Y. Redactable Blockchain Supporting Supervision and Self-Management. In Proceedings of the ASIA CCS, Virtual Event, Hong Kong, China, 7–11 June 2021; pp. 844–858.
37. Chen, X.; Zhang, F.; Kim, K. Chameleon Hashing without Key Exposure. In Proceedings of the International Conference on Information Security, Palo Alto, CA, USA, 27–29 September 2004; pp. 87–98.
38. Camenisch, J.; Derler, D.; Krenn, S. Chameleon Hashes with Ephemeral Trapdoors—And Applications to Invisible Sanitizable Signatures. *Public Key Cryptogr.* **2017**, *20*, 152–182.
39. Ateniese, G.; de Medeiros, B. On the Key Exposure Problem in Chameleon Hashes. In Proceedings of the International Conference on Security in Communication Networks, Amalfi, Italy, 8–10 September 2004; pp. 165–179.
40. Cordoş, A.A.; Bolboacă, S.D.; Prato, R.; Fortunato, F. iGeneration's social media usage in retrieving information related to healthcare education: A web-based survey among Italian and Romanian undergraduate medical students. *Ann. Ist. Super. Sanita* **2019**, *55*, 34–40.
41. Jäntschi, L. Binomial Distributed Data Confidence Interval Calculation: Formulas, Algorithms and Examples. *Symmetry* **2022**, *14*, 1104. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.