Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3923597, 16 pages
https://doi.org/10.1155/2022/3923597

Research Article

A Fine-Grained Medical Data Sharing Scheme with Ciphertext Reencryption

Jiahao Chen[1], **Jingwei Wang**[1], **Xinchun Yin**[1,2] and **Jianting Ning**[3,4]

[1]*College of Information Engineering, Yangzhou University, Yangzhou, China*
[2]*College of Guangling, Yangzhou University, Yangzhou, China*
[3]*College of Computer and Cyberspace Security, Fujian Normal University, Fuzhou, China*
[4]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*

Correspondence should be addressed to Xinchun Yin; xcyin@yzu.edu.cn

Received 9 January 2022; Revised 20 February 2022; Accepted 7 March 2022; Published 27 March 2022

Academic Editor: Liquan Chen

With the fantastic development of cloud computing technology, cloud storage has been widely applied in the field of medical data sharing due to its efficiency and flexibility. Attribute-based encryption (ABE), as a promising technology, plays an important role in the fine-grained sharing of medical data. However, there are two issues: (1) an access policy specified in the encryption phase may need to be updated after a period of time, and (2) the considerable decryption overhead in ABE is too heavy for resource-limited devices. To address these issues, a fine-grained medical data sharing scheme with ciphertext reencryption is proposed. Users with decryption permission can produce reencryption keys and cloud server can reencrypt initial ciphertext. The cloud server can predecrypt ciphertext, which reduces user's decryption overhead. Moreover, the traditional centralized server is replaced with a decentralized blockchain that performs system initialization, key management, and user revocation. We compared the proposed scheme with some existing schemes in terms of function and efficiency; the results show that the efficiency of the proposed scheme outperforms other related schemes. The security analysis shows that the proposed scheme is security and correctness.

1. Introduction

Since cloud computing was proposed, it has received increasing attention [1]. On the one hand, it is an Internet-based value-added service [2], which means that users can obtain cloud services through the network anywhere and anytime. On the other hand, with the continuous development of big data, cloud computing, as a novel and efficient computing paradigm [3], can store and manage huge amounts of data for users. As a result, an increasing number of individuals and enterprises outsource their data to the cloud, promoting the growth of cloud computing.

Cloud storage, a new storage mechanism evolved from cloud computing, can provide users with ubiquitous data storage and access services by centrally processing resources stored in its area. At the same time, by taking use of cloud storage's services, customers not only save costs on local data storage and maintenance but also get access to and share data with greater ease. Therefore, cloud storage is widely used in people's daily life due to its flexibility, convenience, and economy [4].

In the medical field, cloud storage has the advantages of low cost, convenient deployment and efficient storage, and the operations on its platform are basically automated, making it the most ideal environment for deploying standard data security access and sharing mechanisms [5]. On the one hand, the generation, collection, storage, integration, and application of data are decentralized, and cloud storage makes it possible to obtain high-quality medical services through the collection, management, and utilization of these data. On the other hand, with the explosive growth of data and the development of Internet of Things technology, users

can effectively save the cost of local data management and maintenance by outsourcing their massive medical data to the cloud for storage.

In the secure access and sharing mechanism of medical data stored in the cloud, as long as users are authorized, they can obtain safe, convenient, and high-quality medical services in a timely manner by sharing corresponding medical resources in the cloud. This not only greatly improves the storage and sharing efficiency of medical information but also effectively solves the problems of the traditional medical model, such as few medical channels for users, high medical costs for users, and unreasonable medical services for users. Therefore, the secure access and sharing mechanism of cloud storage medical data has become a hot topic in the field of medical and health research [6].

The basic model of secure accessing and sharing mechanism of medical data is shown in Figure 1. In the secure accessing and sharing mechanism of cloud storage medical data, data owners first encrypt the medical data, by paying a fee, the obtained ciphertext can be hosted to the cloud storage center, and the cloud storage center uses access control technology to manage users' permissions to the sharing data.

However, patients' medical data is sensitive and confidential; they may include social security numbers, credit card information, and treatment history. CSP cannot be fully trusted due to various attacks in an untrusted network environment. Once medical data leaked, it may cause serious medical accidents and economic losses, which restricts the application of medical data sharing [7–9]. Therefore, we need to address security and privacy issues when designing practical and secure EHR sharing systems adapted to the cloud environment.

Access control (AC) [10], as a part of the secure accessing and sharing mechanism of medical data, plays a significant role in medical data protection. As in an untrusted cloud storage environment, data owners encrypt the medical data before storing it in the cloud and ensuring that data requesters with decryption capabilities may access the data, therefore realizing the security of medical data cloud storage. Traditional ciphertext AC techniques, however, are no longer suited for cloud-based data sharing due to issues such as low work efficiency, high computing costs, and large bandwidth. This problem has been significantly improved by the proposal of the ciphertext-policy attribute-based encryption (CP-ABE) technique. In CP-ABE, data owners define the AC to ensure that only the data requesters who meet the access policy can decrypt the corresponding ciphertext. This provides not just fine-grained AC but also "one-to-many" encryption. Therefore, with the help of CP-ABE, safe access control and sharing of cloud-storage medical data can be implemented.

In secure accessing and sharing mechanism of medical data based on cloud storage, there are also a large number of scenarios where the ciphertext needs to be reencrypted. For example, ciphertext reencryption technology enables doctor Alice working in P city's hospital to share a patient's medical data with doctor Bob working in Q city's hospital. The ciphertext reencryption technology also enables the doctor Alice, who is on leave, to share her patient's medical data to another doctor Eve in the same hospital. The straightforward solution to reencryption the ciphertext is that Alice first downloads the ciphertext from the cloud, decrypts the ciphertext, and reencrypts the plaintext with the new access policy. After encryption, Alice uploads the new ciphertext to the cloud. When a large amount of patients' ciphertexts need to be reencrypted, this method not only incurs large computing overhead but also increases the communication cost between the cloud and Alice. In order to share data more effectively, proxy reencryption (PRE) [11] is introduced into CP-ABE, which is called ciphertext-policy attribute-based proxy reencryption scheme (CP-ABPRE). In response to the above situation, Alice generates a reencryption key and sends the key to the proxy server. The proxy server can use the key to encrypt the original ciphertexts, so that Bob(Eve) can decrypt the encrypted ciphertext data and realize the sharing of medical data.

In PRE, the ciphertext encrypted under A's public key can be transformed into ciphertext encrypted under B's public key by a semitrusted proxy, and the proxy will not get any information about the underlying encrypted medical data. By implementing the permission of decryption authority, CP-ABPRE can securely and effectively reencrypt ciphertexts and assure the secure access and sharing of medical data. Since the proxy only needs to verify that the attributes in the user-generated reencryption key satisfy the access policy of the original ciphertexts, the proxy server can reencrypt a batch of such ciphertexts. As a result of its great security and low computing overhead, this technology is suitable for a large number of ciphertext reencryption scenarios.

Overall, CP-ABPRE can not only perform one-to-many encryption but it can also safely reencrypt the relevant ciphertexts. It is suitable for the sharing of medical data. For one thing, the proxy can reencrypt the ciphertexts corresponding to access policy A into access policy B without changing the plaintexts. This not only allows for fine-grained access control for medical data users, but it also assures the security of medical data stored in the cloud. In other words, the proxy does not have to decrypt the ciphertext and then reencrypt the plaintext during the whole process. Instead, it may simply complete the ciphertext conversion by directly performing operations on the ciphertexts, which considerably decreases the proxy's computing cost. Therefore, CP-ABPRE is critical for the secure accessing and sharing mechanism of medical data in cloud storage.

However, most of the existing CP-ABPRE schemes must have a trusted center or key generation center (KGC), and the keys are generated and maintained through the KGC. In addition, centralized KGCs are vulnerable to attacks such as a single point of failure, once KGC is compromised by malicious organizations, and it will lead to serious information leakage. Blockchain technology has the potential to tackle this issue effectively. This technology was invented by Nakamoto [12] in 2008, and it has gained global notice. The benefits of blockchain include distributed storage, nontampering, and traceability. As a result, it has a wide range of applications in a variety of industries, including finance, communications, and the Internet of Things. In this paper,

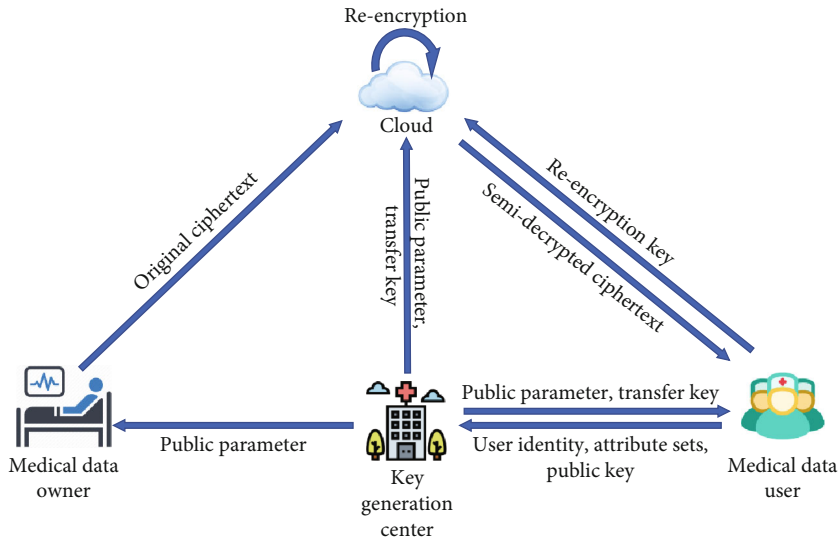

Figure 1: Basic model of secure accessing and sharing mechanism of medical data based on cloud storage.

blockchain is used to replace the traditional trusted center, and users' keys are generated by consensus nodes and users themselves, which solves the problem of centralization of key generation in traditional CP-ABPRE.

2. Related Work

In the traditional public key cryptosystem, the data owner can encrypt the medical data with the public key so that only the data user with the corresponding private key can decrypt ciphertexts, which effectively solves the problem of key distribution. However, due to issues with certificate administration and key escrow, this approach is inapplicable when large-scale medical data owners need to exchange their data. At the same time, in most cases, data owners may not know the particular receivers but aim to accomplish fine-grained access control through strategies created by data owners. As a result, the standard public key cryptosystem is incapable of addressing the issue of data access and sharing in cloud storage.

With the continuous development of cryptographic research, Sahai and Waters [13] prompted an ABE scheme in 2005. Subsequently, Goyal et al. [14] proposed a key-policy attribute-based encryption (KP-ABE) scheme, in which the access policy is bound to the user secret key, and the attribute set is related to the ciphertext. When the attribute set in the ciphertext satisfies the policy in the key, the ciphertext can be decrypted. Bethencourt et al. [15] proposed a ciphertext-policy attribute-based encryption (CP-ABE) scheme. In CP-ABE, the access policy is bound to the ciphertext while the attribute set is related to the user secret key. Only when the attribute set in the key satisfies the access policy in the ciphertext can the ciphertext be decrypted. Since the access policy can be defined by the data owner, CP-ABE is suitable for medical data sharing in cloud storage. In 2011, Waters [16] proposed a CP-ABE scheme that uses linear secret sharing scheme (LSSS). Compared with the scheme of Bethencourt et al. [15], the efficiency has been improved.

In order to address the problem of access policy updating, in 2009, Liang et al. [17] proposed a ciphertext-policy attribute-based proxy reencryption (CP-ABPRE) scheme, which allows a proxy to reencrypt a ciphertext encrypted by policy A to policy B and the proxy cannot obtain any information about the plaintext. However, their scheme cannot resist choose plaintext attacks. To solve this problem, Liang et al. [18] proposed a proxy reencryption scheme based on CP-ABE in 2013, which can resist choose plaintext attacks. However, both the generation of the reencryption key and the reencryption ciphertext require a large amount of calculation. In 2016, Yang et al. [19] employed a cloud server to implement dynamic attribute-based access control, which greatly improved the efficiency of reencryption and reduced the user's decryption overhead. However, the proposed scheme only supports tree access structure. In 2017, Feng et al. [20] prompted an attribute-based proxy reencryption (AB-PRE) scheme which supports LSSS, but it only supports AND gates. In the same year, Ma et al. [21] proposed a verifiable outsourced decryption scheme; the outsourced decryption operation was completed by a decryption server. In 2019, Xu et al. [22] proposed a novel healthcare IoT system fusing advantages of attribute-based encryption, cloud, and edge computing, which provides an efficient, flexible, secure fine-grained access control mechanism with data verification in healthcare IoT network without any secure channel and enables data users to enjoy the lightweight decryption. In 2020, Deng et al. [23] proposed an AB-PRE scheme that supports arbitrary monotonic access policies, but the reencrypted ciphertext only supports identity-based encryption (IBE), and the scalability of reencryption is not as good as ABE. In 2020, Paul et al. [24] proposed an efficient attribute-based proxy reencryption with constant size ciphertext, but his scheme did not support user revocation. In 2021, Xu et al. [25] proposed a practical and secure dynamic EHR sharing system via Cloud. The user

decryption keys of the works above are all generated by the KGC. Once the KGC is compromised, it will cause serious information leakage.

Blockchain was first proposed as the underlying technology of Bitcoin [12]. It is an open distributed public ledger. Due to the hash function and consensus protocol used in the blockchain, it can not only ensure the unforgeability of the data stored in it but also guarantees traceability. Any user can access all the data stored in the ledger, and it is impossible for a malicious user to tamper with this data. In 2017, Huh et al. [26] used public blockchain to manage IoT devices; all the public key of users are stored in blockchain. In 2017, Liu et al. [27] used private blockchain to ensure data integrity. In 2019, Hu et al. [28] proposed a blockchain-based data sharing scheme, a decentralized system is established based on smart contracts.

2.1. Contributions. In this paper, we proposed a blockchain-aided fine-grained medical data sharing scheme supporting ciphertext reencryption. The characteristics of the scheme are as follows.

2.1.1. Blockchain-Aided ABE. The proposed scheme allows users to generate private keys by themselves. This prevents the usage of users' private keys created by KGC in traditional schemes. The KGC in traditional ABE is replaced by the consensus nodes of the distributed consensus blockchain, and the immutable ledger can store transactions. The user's public key and system public parameters can be uploaded to the blockchain to ensure that the data is not tampered with.

2.1.2. Ciphertext Reencryption. The proposed scheme allows the cloud to reencrypt a ciphertext decryptable by user A into a ciphertext decryptable by user B without disclosing the plaintext or the authorizer's private key. The complete procedure does not necessitate decryption.

2.1.3. Collusion Resistance. The proposed scheme adopts the user's Global Identifier (GID) to bind attributes of users. Only attributes belonging to the same user can decrypt the ciphertext, thereby preventing the system from colluding attacks.

2.1.4. Lightweight Decryption and Verification. The proposed scheme allows the cloud to pre-decrypt the ciphertext, lowering the user's computational cost in the decryption process. Data user can decrypt the final ciphertext with only one exponentiation computation, and the user can verify if the cloud has made the correct transformation.

2.1.5. Efficient User Revocation. For situations such as the resignation of doctors and the recovery of patients, their access rights need to be revoked. The proposed scheme supports efficient user revocation.

2.2. Paper Organization. Section 3 introduces the background knowledge. The system architecture model is formally defined in Section 4. Details of the proposed scheme are depicted in Section 5. Section 6 introduces the performance analysis. Ultimately, the conclusion is given in Section 7.

3. Preliminaries

3.1. Notations. Related notations and definitions used in the proposed scheme are presented in Table 1.

3.2. Access Structures. Let $\{P_1, P_2, \cdots, P_n\}$ be the set of n participants. For set $A \subseteq 2^{\{P_1, P_2, \cdots P_n\}}$, if $\forall B, C \subseteq \{P_1, P_2, \cdots P_n\}$, $B \in C$, $B \subseteq C \Rightarrow C \in A$, then A is said to be monotonic. Among them, the set belonging to A is called authorized set; otherwise, it is called unauthorized set [29].

3.3. Bilinear Maps. Let G and G_T be two multiplicative cyclic groups of order p. g is a generator of G and $e : G \times G \longrightarrow G_T$ is a map with three properties: (1) bilinearity—for $\forall u, v \in G$ and $\forall a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$; (2) non-degeneracy—$e(g, g) \neq 1$; and (3) computability—for all $\forall u, v \in G$, there exists effective algorithms to calculate $e(u, v)$.

3.4. Linear Secret Sharing Schemes. Let A denote the attribute university and p denote a prime number. The secret sharing scheme Π on the secret space Z_p realizes the access policy on A. If Π satisfies the following two properties, then, Π is linear [30].

(1) Each part of the secret $s \in Z_p$ corresponds to an attribute in A, and each part constitutes a vector on Z_p

(2) For the access policy $S = (M, \rho)$, M is a $l \times n$ secret sharing matrix. The function ρ maps the $i_{th} \in \{1, 2, \cdots, l\}$ row of M to an attribute $\rho(i)$ of the attribute set A. For example, construct a column vector $u = (s, y_2, y_3, \cdots, y_n)^{\mathrm{T}}$, where $y_2, y_3, \cdots, y_n \in Z_p$ are random numbers used to hide the secret value s. Then, $M_u \in Z_p^{l \times 1}$ is a vector with l rows and 1 column. That is, the secret value s is divided into l parts according to Π. $(Mu)_i$ corresponds to the attribute $\rho(i)$, where $i \in [l]$

Every LSSS has linear reconstruction property. Suppose that Π is a LSSS for the access structure S. $P \in S$ is the set of arbitrary authorization, $I = \{i \in [l] \wedge \rho(i) \in P\}$ and $I \subseteq \{1, 2, \cdots, l\}$. For any legal sharing of $\{\lambda_i = (Mu)_i\}_{i \in I}$, there exists a set of constants $\{\sigma_i \in Z_p\}_{i \in I}$ that can be calculated in polynomial time. For the unauthorized set p', there is no such constant set as $\{\sigma_i\}_{i \in I}$.

3.5. Blockchain Technology. Blockchain is a specific data structure based on transparent and trusted consensus rules in a peer-to-peer network. It combines data blocks in a chronological order to form a specific data structure. Blockchain cryptographically guarantees a decentralized and credible distributed shared ledger system whose data cannot be tampered with, forged or traced. Blockchain is mainly divided into three types: public blockchain, consortium blockchain, and private blockchain. The consortium blockchain usually designates one or more preselected consensus nodes as the bookkeeper. The generation of each block is jointly determined by all consensus nodes. The connected nodes can participate in activities on the chain but do not care about the process of accounting. The public blockchain adopts a proof

Table 1: Notations and definitions used in the proposed scheme.

Notation	Definition
p	Large prime
G, G_T	Cyclic group of order p
g	Generator of G
Z_p	Integer domain not greater than p
e	Bilinear map
(M, ρ)	Access policy(M is a matrix of $l \times n$, ρ is a map)
CSP	Cloud server provider
MDO	Medical data owner
MDU	Medical data user
λ	Security parameter
A	Attribute universe
CN	Consensus nodes
H	Hash function
SE	Symmetric-key encryption algorithm
GID	User's Global Identifier

of work mechanism to exchange efficiency for fairness. The private blockchain sets a single central accounting node to exchange fairness for efficiency. Compared with the public blockchain and private blockchain, the participants in the consortium blockchain know each other's digital identities and can reach business consensus in advance. The proof-of-work mechanism is no longer needed; thus, this can reduce energy consumption and improve efficiency. Since the consortium blockchain takes into account both efficiency and fairness, it is considered by many researchers to be the most suitable blockchain for medical data sharing application scenarios.

3.6. Random Oracle Model. The random oracle model (ROM) was proposed in 1993 [31]; the ROM uses the character of a random oracle to show the security of the algorithm. The random oracle can be considered a black box in theory, and the black box is embedded with a certain algorithm so that the inquirer cannot obtain any information no matter how many inquiries are made. Among them, the returned value is collision resistant, the value is the same for the same query, and the value is unpredictable for different queries. The random oracle $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ is a specific type of hash function, and the function satisfies the following characteristics:

(1) Uniformity: given the result y after H operation, the distribution of y on $\{0,1\}^n$ is uniform.

(2) Effectiveness: given a string x, H can do certain operations to get $H(x)$. $H(x)$ is the effective output of H in low-order polynomial time of x length.

(3) Certainty: if the two numbers input by the oracle H are the same, then the two corresponding outputs must be the same.

3.7. q-DBPBDHE2 Assumption. The system selects two multiplicative cyclic groups G and G_T whose order is prime p, g is the generator of group G, and $e : G \times G \longrightarrow G_T$ is a bilinear mapping. The system randomly selects $a, s, b_1, b_q \in Z_p^*$ and $R \in G_T$. The system defines a vector $D = (p, g, G, e, g^s, \{g^{a^i}\}_{i\in[2q],i\neq q+1}, \{g^{b_j a^i}\}_{(i,j)\in[2q,q],i\neq q+1}, \{g^{s/b_i}\}_{i\in[q]}, \{g^{sb_j a^i/b_{j'}}\}_{(i,j,j')\in[q+1,q,q],j\neq j'})$. If the adversary cannot distinguish a random factor $R \in G_T$ from $e(g,g)^{a^{q+1}s}$ with a nonnegligible advantage ε in probabilistic polynomial time (PPT), then q-DBPBDHE2 assumption establish. Advantage ε is defined as $|\Pr[A(D, e(g,g)^{a^{q+1}s}) = 0] - \Pr[A(D, R) = 0]| \geq \varepsilon$.

4. System Architecture

4.1. Functionalities of Each Entity. The proposed scheme includes 4 entities. As shown in Figure 2, they are consensus node (CN), cloud server provider (CSP), medical data owner (MDO), and medical data user (MDU).

4.1.1. Consensus Node. Consensus nodes are the maintainer of the consortium blockchain. Each consensus node manages the attributes of MDUs, initializing the system, generating transfer keys for MDU, and maintaining a revocation list.

4.1.2. Cloud Server Provider. CSP is responsible for storing ciphertext and managing the attribute transfer keys for each user. CSP receives the reencryption key and generates the corresponding reencryption ciphertext. CSP also performs outsourcing decryption for MDU. Furthermore, CSP needs to revoke illegal users in the system.

4.1.3. Medical Data Owner. MDO first encrypts medical data with a symmetric key encryption algorithm, then encrypts the symmetric-key with a CP-ABPRE, and finally uploads the ciphertext to the CSP.

4.1.4. Medical Data User. MDU with sufficient attributes can request the CSP to perform the semidecryption algorithm and then decrypt the semidecrypted ciphertext with their own private key efficiently. MDU also generates a reencryption key and uploads it to the CSP. It is worth noting that the MDU can only obtain the semidecrypted ciphertext after outsourcing decryption of the CSP but cannot obtain the original ciphertext in the CSP.

4.2. Formal Definition

(1) GlobalSetup$(\lambda, A) \longrightarrow (\mathrm{PP}, R)$: the algorithm inputs the security parameters λ according to the security requirements of the scheme and attribute universe A and outputs the system public parameter PP, an empty revocation list R.

(2) NodeSetup$(\mathrm{PP}, \eta) \longrightarrow (\mathrm{NPK}_\eta, \mathrm{NSK}_\eta)$: this algorithm takes the public parameters PP and CN's identity η as input and outputs CN's node public key NPK_η and node secret key NSK_η.

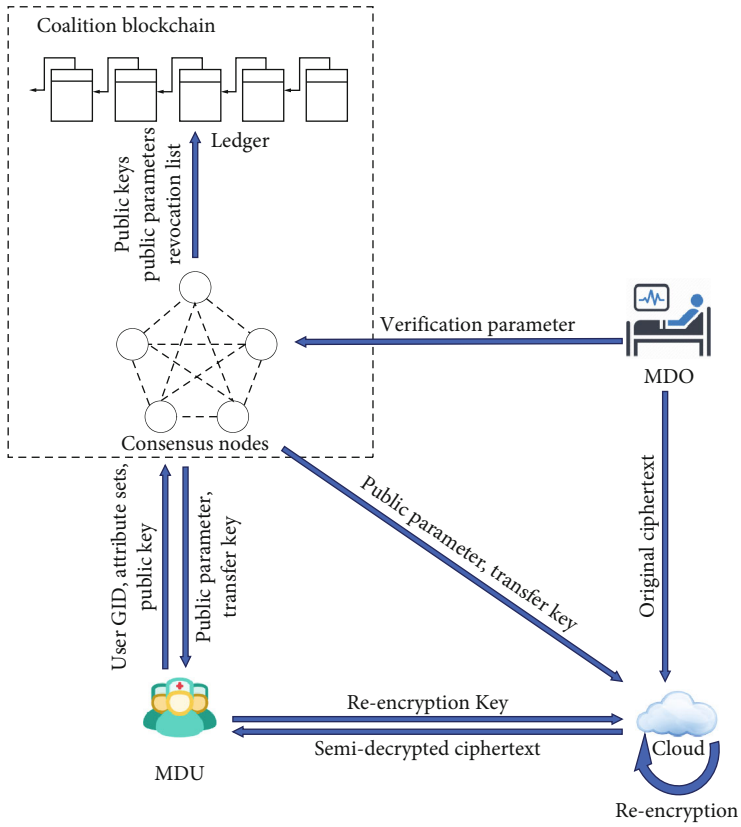

Figure 2: System model.

(3) UserKeyGen(PP, GID) $\longrightarrow$ (UPK, USK): the algorithm inputs the PP and user GID and outputs the user public key UPK and user private key USK.

(4) TKGen(PP, GID, S, NSK_{η}, UPK) $\longrightarrow$ ($\mathrm{TK}_{S,\mathrm{GID}}$): the algorithm inputs the public parameter PP, user GID, user attribute set S, node secret key NSK_{η} and user public key UPK and outputs the user's transfer key $\mathrm{TK}_{S,\mathrm{GID}}$. CN sends it to the user and CSP; CSP adds (GID, $\mathrm{TK}_{S,\mathrm{GID}}$) to the key list KT.

(5) Enc(PP, m, (M, ρ), NPK_{η}) $\longrightarrow$ (CT): this algorithm inputs the public parameter PP, a EHR plaintext m, an access policy (M, ρ), and node public key NPK_{η} and outputs the ciphertext CT.

(6) ReKeyGen(PP, $\mathrm{TK}_{S,}$ GID, USK, NPK_{η}, S, (M', ρ')) $\longrightarrow$ $\mathrm{ReKey}_{(S,\mathrm{GID})\longrightarrow}$ (M',ρ'): this algorithm inputs the public parameter PP, transfer key $\mathrm{TK}_{S,\mathrm{GID}}$, user private key USK, node public key NPK_{η}, user attribute set S, and a new access policy (M', ρ') and outputs the reencryption key $\mathrm{ReKey}_{(S,\mathrm{GID})\longrightarrow(M',\rho')}$.

(7) ReEnc(PP, $\mathrm{ReKey}_{(S,\mathrm{GID})\longrightarrow(}$ $M',\rho')$, CT $)\longrightarrow$ $C^{R}_{(M',\rho')}$: this algorithm inputs the public parameter PP, reencryption key $\mathrm{ReKey}_{(S,\mathrm{GID})\longrightarrow(M',\rho')}$, and ciphertext CT and outputs the reencryption ciphertext $C^{R}_{(M',\rho')}$.

(8) TransDec(PP, CT, S, $\mathrm{TK}_{S,\mathrm{GID}}$, UPK) $\longrightarrow$ $\mathrm{CT}_{\mathrm{GID}}$: this algorithm inputs the public parameter PP, ciphertext CT, user attribute set S, transform key $\mathrm{TK}_{S,\mathrm{GID}}$, and user public key UPK and outputs the semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}}$.

(9) UserDec(USK, $\mathrm{CT}_{\mathrm{GID}}$) $\longrightarrow$ m: this algorithm inputs the user private key USK and outputs the plaintext m.

(10) $\mathrm{TransDec}_{R}$(PP, $C^{R}_{(M',\rho')}$, S', $\mathrm{TK}_{S',\mathrm{GID}}$) $\longrightarrow$ $\mathrm{CT}_{\mathrm{GID}'}$: this algorithm inputs the public parameter PP, ciphertext $C^{R}_{(M',\rho')}$, user attribute set S', and transform key $\mathrm{TK}_{S',\mathrm{GID}}$ and outputs the semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}'}$.

(11) $\mathrm{UserDec}_{R}$(PP, USK, $\mathrm{CT}_{\mathrm{GID}'}$) $\longrightarrow$ m: this algorithm inputs the public parameter PP, user private key USK, and semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}'}$ and outputs the plaintext m

(12) Revoke(GID, R, KT) $\longrightarrow$ (R, KT/(GID, $\mathrm{TK}_{S,\mathrm{GID}}$)): this algorithm inputs the identity of the revoked user GID, revocation list R, and key list KT and

outputs an updated revocation list R' and an updated key list KT/(GID, $TK_{S,GID}$).

4.3. Threat Model. In the system, consensus nodes and MDO are fully trusted. Each consensus node manages the attributes of MDUs, initializing the system, generating transfer keys for MDU, and maintaining a revocation list. MDO honestly encrypts medical data and generates ciphertext according to its own wishes and uploads it to CSP.

The CSP is semitrusted. CSP is semitrusted, and it follows our scheme but tries to learn sensitive data without any active attack. In addition to this, CSP may also dishonestly decrypt ciphertext for MDU to ease the burden of data storage or for any other reason.

MDU are untrusted. MDU try to learn about unauthorized data and even try to collude with other MDU to gain access to sensitive data. To prevent attacks from untrusted DUs, we apply cryptosystems to ensure data confidentiality.

From the above, the security of our proposed scheme depends on the blockchain and cryptosystem. As we all know, all data stored on the blockchain cannot be tampered with. The cryptosystems in our scheme are hash functions, symmetric encryption, and our scheme. We apply a collision-resistant hash function to prevent adversaries from compromising data integrity. Symmetric encryption is known to be resistant to all kinds of attacks. For the security of our scheme, we propose a formal security model in the next subsection to demonstrate the security of our scheme.

4.4. Security Model. The security model of the proposed scheme is a static security model. The static security model satisfies the confidentiality of messages; it can be defined by a security game between an adversary Adv and a challenger *C*. The game is performed in the form of query-answer between Adv and *C*. All inquiries must be completed before the challenge phase. Since static security model of the scheme is mainly aimed at the collusion attack of multiple legitimate MDUs, it is required that Adv can ask the MDU for the private key many times and ask other legitimate MDUs: GID to partially decrypt the ciphertext CT_{GID}. The implementation of the latter is based on that Adv can ask CSP for the corresponding transform key, so that Adv can obtain the GID's CT_{GID} according to the transform key of CSP.

Definition 1. Given a probabilistic polynomial time, if there is no one adversary Adv can win the following game with a nonnegligible advantage, the proposed scheme is static security. In the game, A represents attribute universe of the protocol, and θ represents the consensus node set. Assume that each attribute is assigned to a consensus node (although each consensus node can manage multiple properties).

The game consists of the following 4 phases.

(1) Initialization phase: the initialization phase is performed by challenge *C*. *C* inputs λ according to security requirements. Then, the GlobalSetup algorithm is performed, and the public parameter PP is sent to Adv.

(2) Query phase: define consensus nodes as N_θ. Adv does the following query to *C*.

(i) Query 1: Adv chooses some N_θ; Adv asks *C* for their corresponding public key NPK_θ.

(ii) Query 2: Adv selects some MDUs' $\{GID_i\}_{i=1}^m$ and asks for their key pairs $\{UPK_i, USK_i\}_{i=1}^m$.

(iii) Query 3: Adv asks *C* for the transfer key: input MDUs' $GID_i (1 \le i \le n)$ and their attribute sets $S_i \in A$; *C* performs $TKGen(PP, GID, S, NSK_\theta, UPK) \longrightarrow (TK_{S,GID})$ to generate transfer key $TK_{S,GID}$. Then, *C* returns $TK_{S,GID}$ to Adv. At the same time, the query made by Adv includes not only the query to obtain the transfer key corresponding to the MDUs' GID_i but also the query to obtain the transfer key corresponding to other MDUs (that is, if n is larger than m).

(iv) Query 4: Adv asks *C* for reencryption key: input user $GID_i (1 \le i \le n)$, user attribute sets $S_i \in A$, and access policy (M', ρ'); *C* performs $ReKeyGen(PP, TK_{S,GID}, USK, NPK_\eta, S, (M', \rho')) \longrightarrow ReKey_{(S,GID)\longrightarrow(M',\rho')}$ to generate reencryption key $ReKey_{(S,GID)\longrightarrow(M',\rho')}$. Then, *C* returns $ReKey_{(S,GID)\longrightarrow(M',\rho')}$ to Adv. $TK_{S,GID}$ is generated by TKGen() algorithm.

(3) Challenge phase: Adv submits two equal-length plaintexts $M_0{}^*$, $M_1{}^*$, and an access policy (M^*, ρ^*) to *C*. *C* randomly selects $b \in \{0, 1\}$ and executes Enc() algorithm to calculate challenge ciphertext and return it to *Adv*. The limitation is that the $\{GID_i\} (1 \le i \le m)$ of the user who asked for the private key during the query; their attribute set S_i does not satisfy (M^*, ρ^*).

(4) Guess phase: Adv outputs guess $b' \in \{0, 1\}$ on b. If $b = b'$, Adv wins the game. The advantage of Adv winning is defined as $|\Pr[b = b'] - 1/2|$.

5. Proposed Scheme

5.1. Concrete Construction. This section will give the design of the protocol. To better understand the protocol, the access policy of the protocol is represented by a two-tuple (M, ρ), where M is a matrix of $l \times n$ and ρ maps each row of M to an attribute $\rho_{(x)}$. At the same time, define a function $F : A \longrightarrow A_\eta$; A represents attribute universe of the protocol, and A_η represents attributes managed by a consensus node (CN) with identity η: CN_η; this function can implement the mapping of the attribute $u \in A$ to CN_η. The

corresponding function $\delta(\bullet) = F(\rho(\bullet))$can realize the mapping of the rows of the matrix to a consensus node. The concrete construction is as follows:

(1) GlobalSetup$(\lambda, A) \longrightarrow (\mathrm{PP}, R)$: the GlobalSetup algorithm is performed by all CNs. They take a security parameter λ and system attribute universe A as input and outputs system public parameter PP, an empty revocation list R. The algorithm randomly selects a security parameter λ. Let G and G_T be two multiplicative cyclic groups with prime order p, g be a generator of G, $\mathscr{K}$ be the key space of symmetric encryption, and $e: G \times G \longrightarrow G_T$ be a nondegenerate bilinear pair. The algorithm selects 4 safety hash functions: $H_1: \{0,1\}^* \longrightarrow G$, $H_2: G_T \longrightarrow Z_p^*$, $H_3: A \longrightarrow G$, and $H_4: \{0,1\}^* \longrightarrow \mathscr{K}$, and a symmetric-key encryption algorithm: $\mathrm{SE} = (\mathrm{SE.Enc}(\bullet), \mathrm{SE.Dec}(\bullet))$. In SE, SE.Enc$(\bullet)$ is a symmetric encryption algorithm. At last, CNs output global public parameters $\mathrm{PP} = \{p, G, G_T, g, e, H_1, H_2, H_3, H_4, \mathrm{SE}, A, A_\eta\}$, an empty revocation list R, and sends them to the blockchain.

(2) NodeSetup$(\mathrm{PP}, \eta) \longrightarrow (\mathrm{NPK}_\eta, \mathrm{NSK}_\eta)$: the NodeSetup algorithm is performed by all CNs. Each CN_η takes public parameters PP and identity η as input, output node public key NPK_η, and node secret key NSK_η. Each CN_η randomly selects α_η, $y_\eta \in Z_p^*$ and computes $e(g,g)^{\alpha_\eta}$, g^{α_η}, g^{y_η}. They keep $\mathrm{NSK}_\eta = \{\alpha_\eta, y_\eta\}$ confidential and publish $\mathrm{NPK}_\eta = \{e(g,g)^{\alpha_\eta}, g^{\alpha_\eta}, g^{y_\eta}\}$.

(3) UserKeyGen$(\mathrm{PP}, \mathrm{GID}) \longrightarrow (\mathrm{UPK}, \mathrm{USK})$: MDU performs the UserKeyGen algorithm. The algorithm inputs PP and user GID and outputs MDU's user public key UPK and user private key USK. The algorithm randomly selects $x_{\mathrm{GID}} \in Z_p^*$ and computes $K_1 = g^{x_{\mathrm{GID}}}$, $K_2 = H_1(\mathrm{GID})^{x_{\mathrm{GID}}}$. It sets $\mathrm{UPK} = (K_1, K_2)$ and $\mathrm{USK} = x_{\mathrm{GID}}$.

(4) TKGen$(\mathrm{PP}, \mathrm{GID}, S, \mathrm{NSK}_\eta, \mathrm{UPK}) \longrightarrow (\mathrm{TK}_{S,\mathrm{GID}})$: CN performs the TKGen algorithm. The algorithm inputs public parameter PP, user GID, user attribute set S, node secret key NSK_η, and user public key UPK and outputs transfer key $\mathrm{TK}_{S,\mathrm{GID}}$. For all attributes, $u \in A$, if u is managed by CN_η. Then CN_η randomly selects $t_u \in Z_p^*$ and computes $\mathrm{CSK}_{u,\mathrm{GID}} = g^{x_{\mathrm{GID}}\alpha_\eta} \bullet H_1(\mathrm{GID})^{x_{\mathrm{GID}} y_\eta} \bullet H_3(u)^{t_u}$ and $\mathrm{CSK}'_{u,\mathrm{GID}} = g^{t_u}$. Then, CN_η sets $\mathrm{TK}_{S,\mathrm{GID}} = (\mathrm{CSK}_{u,\mathrm{GID}}, \mathrm{CSK}'_{u,\mathrm{GID}})_{u \in S}$ and sends it to MDU and CSP. CSP adds $(\mathrm{GID}, \mathrm{TK}_{S,\mathrm{GID}})$ to the key list KT.

(5) Enc$(\mathrm{PP}, m, (M, \rho), \mathrm{NPK}_\eta) \longrightarrow (\mathrm{CT})$: MDO performs the Enc algorithm. Let M be a $l \times n$ matrix and ρ be the function that maps each row of M to an attribute. The algorithm randomly selects $Y \in G_T$ and sets $\mathrm{key} = H_4(Y)$, selects a positive integer ϖ, and concatenates ϖ bit 0 string after the message m, which is used for outsourced decryption verification. The algorithm uses the SE.Enc algorithm to obtain the ciphertext of the message m: $M_{\mathrm{SE}} = \mathrm{SE.Enc}_{\mathrm{key}}(m \| 0^\varpi)$, where $\|$ denotes concatenation of a string. Then, it computes $\mathrm{Ver} = H_1(m \| 0^\varpi)$.The algorithm randomly selects $s, y_2, \cdots, y_n, z_2, \cdots, z_n \in Z_p^*$ and sets vector $v = (s, y_2, \cdots y_n)^\intercal$, $\omega = (0, z_2, \cdots, z_n)^\intercal$. For $x \in [l]$, the algorithm randomly selects $Q_x, R_x, r_x \in Z_p^*$ and computes $C_0 = Y \bullet e(g,g)^s$, $C_{1,x} = g^{Q_x} g^{\alpha_{\delta(x)} r_x}$, $C_{2,x} = g^{-r_x}$, $C_{3,x} = g^{y_{\delta(x)} r_x} g^{R_x}$, $C_{4,x} = H_3(\rho(x))^{r_x}$, $\lambda_x = (Mv)_x$, $\omega_x = (M\omega)_x$, $C_{5,x} = \lambda_x - Q_x$, and $C_{6,x} = \omega_x - R_x$. The algorithm uploads $\mathrm{CT} = ((M, \rho), C_0, \{C_{1,x}, C_{2,x}, C_{3,x}, C_{4,x}, C_{5,x}, C_{6,x}\}_{x \in [l]}, M_{\mathrm{SE}})$ to CSP and uploads Ver to blockchain.

(6) ReKeyGen$(\mathrm{PP}, \mathrm{TK}_{S,\mathrm{GID}}, \mathrm{USK}, \mathrm{NPK}_\eta, S, (M', \rho')) \longrightarrow \mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$: MDU performs the ReKeyGen algorithm. It inputs public parameter PP, transfer key $\mathrm{TK}_{S,\mathrm{GID}}$, user private key USK, node public key NPK_η, user attribute set S, and a new access policy (M', ρ') and outputs reencryption key $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$. The algorithm randomly selects $s', y'_2, \cdots, y'_n, z'_2, \cdots, z'_n, r'_x \in Z_p^*, \beta \in G_T$ and sets two vectors $v' = (s', y'_2, \cdots, y'_n)^\intercal$ and $\omega' = (0, z'_2, \cdots, z'_n)^\intercal$. Then, for $x \in [l']$, the algorithm sets $\lambda'_x = (M'v')_x$ and $\omega'_x = (M'\omega')_x$. The algorithm computes $C'_0 = \beta \bullet e(g,g)^{s'}$, $C'_{1,x} = g^{\lambda_x'} g^{\alpha_{\delta'(x)} r'_x}$, $C'_{2,x} = g^{-r_x'}$, $C'_{3,x} = g^{y_{\delta'(x)} r'_x} g^{\omega_x'}$, and $C'_{4,x} = H_3(\rho'(x))^{r'_x}$ and then sets $rk_1 = ((\mathrm{CSK}_{u,\mathrm{GID}})^{(x_{\mathrm{GID}})^{-1}})^{H_2(\beta)}$, $rk_2 = ((\mathrm{CSK}'_{u,\mathrm{GID}})^{(x_{\mathrm{GID}})^{-1}})^{H_2(\beta)}$, $rk_3 = g^{H_2(\beta)}$, $rk_4 = (H_1(\mathrm{GID})^{x_{\mathrm{GID}}})^{(x_{\mathrm{GID}})^{-1} H_2(\beta)}$, and $rk_5 = ((M', \rho'), C'_0, \{C'_{1,x}, C'_{2,x}, C'_{3,x}, C'_{4,x}\}_{x \in [l']})$. Finally, the algorithm generates $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')} = (S, rk_1, rk_2, rk_3, rk_4, rk_5)$ and sends $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$ to CSP.

(7) ReEnc$(\mathrm{PP}, \mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}, \mathrm{CT}) \longrightarrow C^R_{(M', \rho')}$: CSP performs the ReEnc algorithm. It inputs public parameter PP, reencryption key $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$, and ciphertext CT and outputs reencryption ciphertext $C^R_{(M', \rho')}$. After receiving $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$ sent by the MDU, the algorithm will check whether the attribute set S in $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$ matches the access policy (M, ρ) in CT. If S does not match (M, ρ), it indicates $\mathrm{ReKey}_{(S,\mathrm{GID}) \longrightarrow (M', \rho')}$ is illegal, and CSP terminates the reencryption operation. If S matches (M, ρ), the algorithm performs the following steps to compute the reencrypted ciphertext $C^R_{(M', \rho')}$.

The algorithm sets $I = \{x : \rho(x) \in S\} \subseteq (1, 2, \cdots, l)$ and computes constant $\{c_x \in Z_p^*\}$ to let $\sum_{x\in I} c_x M_x = (1, 0, .., 0)$. Then, the algorithm computes $C_4 = \prod_{x\in I}(e(C_{1,x} g^{C_{5,x}}, rk_3))^{c_x}$ and $C_5 = \prod_{x\in I}(e(rk_1, C_{2,x}) e(rk_4, C_{3,x} g^{C_{6,x}}) e(rk_2, C_{4,x}))^{c_x}$ and then computes $C_6 = C_4 \bullet C_5 = e(g, g)^{sH_2(\beta)}$. Finally, the algorithm outputs $C^R_{(M',\rho')} = ((M', \rho'), C_0, C_6, rk_5, M_{\mathrm{SE}})$.

(8) $\mathrm{TransDec}(\mathrm{PP}, \mathrm{CT}, S, \mathrm{TK}_{S,\mathrm{GID}}, \mathrm{UPK}) \longrightarrow \mathrm{CT}_{\mathrm{GID}}$: CSP performs the TransDec algorithm. It inputs public parameter PP, ciphertext CT, user attribute set S, transform key $\mathrm{TK}_{S,\mathrm{GID}}$, and user public key UPK and outputs semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}}$. The algorithm checks whether S matches (M, ρ) in CT; if S does not match (M, ρ), it aborts; otherwise, the algorithm lets $I = \{x : \rho(x) \in S\} \subseteq \{1, 2, \cdots, l\}$ and computes $\{c_x \in Z_p^*\}$ to let $\sum_{x\in I} c_x M_x = (1, 0, .., 0)$. Thus, $\sum_{x\in I} \lambda_x c_x = s$, $\sum_{x\in I} \omega_x c_x = 0$. The algorithm computes $C_{1,\mathrm{GID}} = \prod_{x\in I} e(C_{1,x} g^{C_{5,x}}, g)^{c_x}$ and $C_{2,\mathrm{GID}} = \prod_{x\in I} (e(\mathrm{CSK}_{\rho(x),\mathrm{GID}}, C_{2,x}) \times e(H_1(\mathrm{GID})^{x_{\mathrm{GID}}}, C_{3,x} g^{C_{6,x}}) e(\mathrm{CSK}'_{\rho(x),\mathrm{GID}}, C_{4,x}))^{c_x}$. The algorithm sends semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}} = (C_0, C_{1,\mathrm{GID}}, C_{2,\mathrm{GID}}, M_{SE})$ to MDU.

(9) $\mathrm{UserDec}(\mathrm{USK}, \mathrm{CT}_{\mathrm{GID}}) \longrightarrow m$: this algorithm is performed by MDU; MDU uses USK to decrypt $\mathrm{CT}_{\mathrm{GID}}$ and obtains m. The algorithm computes $C_{1,GID}(C_{2,GID})^{1/x_{\mathrm{GID}}} = e(g, g)^s$. It recovers $Y = C_0/e(g, g)^s$ and computes key $= H_4(Y)$, $m' = \mathrm{SE.Dec}_{\mathrm{key}}(M_{\mathrm{SE}})$. The algorithm checks if there is a redundancy 0^ω appended after m'; if it is, $m' = m \| 0^\omega$. Then, it checks if $H_1(m') = \mathrm{Ver}$ (Ver can be obtained in blockchain); if it is, the message m can be obtained by truncating ω-bit 0 string. Otherwise, CSP is dishonest to return an incorrect semidecrypted ciphertext, and algorithm outputs $\perp$.

(10) $\mathrm{TransDec}_R(\mathrm{PP}, C^R_{(M',\rho')}, S', \mathrm{TK}_{S',\mathrm{GID}}) \longrightarrow \mathrm{CT}_{\mathrm{GID}'}$: CSP performs the $\mathrm{TransDec}_R$ algorithm. It inputs public parameter PP, ciphertext $C^R_{(M',\rho')}$, user attribute set S', and transform key $\mathrm{TK}_{S',\mathrm{GID}}$ and outputs semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}'}$. The algorithm checks whether S' matches (M', ρ') in $C^R_{(M',\rho')}$; if S' does not match (M', ρ'), it aborts. Otherwise, the algorithm performs the following steps: CSP let $I' = \{x : \rho'(x) \in S'\} \subseteq \{1, 2, \cdots, l'\}$ and computes $\{c'_x \in Z_p^*\}$ to let $\sum_{x\in I'} c'_x M'_x = (1, 0, .., 0)$. It then computes $C_{1,\mathrm{GID}'} = \prod_{x\in I'} e(C'_{1,x}, g)^{c'_x}$ and $C_{2,\mathrm{GID}'} = \prod_{x\in I'} (e \quad (\mathrm{CSK}_{\rho'(x),\mathrm{GID}'}, C'_{2,x}) e(H_1(\mathrm{GID}')^{x_{\mathrm{GID}'}}, C'_{3,x}) \times e(\mathrm{CSK}'_{\rho'(x),\mathrm{GID}}, C'_{4,x}))^{c'_x}$. The algorithm sends semidecrypted ciphertext $\mathrm{CT}_{\mathrm{GID}'} = (C_0, C'_0, C_6, C_{1,\mathrm{GID}'}, C_{2,\mathrm{GID}'}, M_{\mathrm{SE}})$ to MDU.

(11) $\mathrm{UserDec}_{\mathrm{R}}(\mathrm{PP}, \mathrm{USK}, \mathrm{CT}_{\mathrm{GID}'}) \longrightarrow m$: This algorithm is performed by MDU; it inputs PP, USK, $\mathrm{CT}_{\mathrm{GID}'}$ to decrypt $\mathrm{CT}_{\mathrm{GID}'}$ to obtain m. The algorithm uses USK to compute $\beta = \beta \bullet e(g, g)^{s'}/e(g, g)^{s'}$ and then recovers $Y = C_0/(C_6)^{1/H_2(\beta)}$. MDU computes key $= H_4(Y)$ and $m' = \mathrm{SE.Dec}_{\mathrm{key}}(M_{\mathrm{SE}})$. The algorithm then checks if there is a redundancy 0^ω appended after m', if it is, $m' = m \| 0^\omega$. Then, it checks if $H_1(m') = \mathrm{Ver}$ (Ver can be obtained in blockchain), if it is, the message m can be acquired by omitting ω-bit 0 string. If not, CSP is dishonest to return an incorrect semidecrypted ciphertext and the algorithm outputs $\perp$.

(12) $\mathrm{Revoke}(\mathrm{GID}, R, \mathrm{KT}) \longrightarrow (R', \mathrm{KT}/(\mathrm{GID}, \mathrm{TK}_{S,\mathrm{GID}}))$: the algorithm is performed by CN and CSP; CN first updates R and uploads the latest R' to blockchain. CSP checks the latest R' in blockchain and inputs malicious user's GID in R' and KT and outputs an updated $\mathrm{KT}/(\mathrm{GID}, \mathrm{TK}_{S,\mathrm{GID}})$. If the user's GID needs to be revoked, CSP will find $\{\mathrm{GID}, \mathrm{TK}_{S,\mathrm{GID}}\}$ in KT and delete $\{\mathrm{GID}, \mathrm{TK}_{S,\mathrm{GID}}\}$ in KT.

5.2. Correctness

5.2.1. Correctness of Decryption of Original Ciphertext CT. The CSP checks whether S matches (M, ρ) in CT; if S does not match (M, ρ), it aborts; otherwise, the algorithm lets $I = \{x : \rho(x) \in S\} \subseteq \{1, 2, \cdots, l\}$ and computes $\{c_x \in Z_p^*\}$ to let $\sum_{x\in I} c_x M_x = (1, 0, .., 0)$. Thus, $\sum_{x\in I} \lambda_x c_x = s$, $\sum_{x\in I} \omega_x c_x = 0$. There is $\eta = \delta(x) = F(\rho(x))$. CSP computes formula (3) to acquire $C_{1,\mathrm{GID}}$.

$$\begin{aligned} C_{1,\mathrm{GID}} &= \prod_{x\in I} e\left(C_{1,x} g^{C_{5,x}}, g\right)^{c_x} = \prod_{x\in I} e\left(g^{Q_x} g^{\alpha_{\delta(x)} r_x} g^{\lambda_x - Q_x}, g\right)^{c_x} \\ &= \prod_{x\in I} e\left(g^{\alpha_{\delta(x)} r_x} g^{\lambda_x}, g\right)^{c_x} = \prod_{x\in I} e(g, g)^{\lambda_x c_x} e(g, g)^{\alpha_{\delta(x)} r_x c_x} \\ &= e(g, g)^{s + \sum_{x\in I} \alpha_{\delta(x)} r_x c_x}. \end{aligned} \tag{1}$$

CSP then computes formula (4) to acquire $C_{2,\mathrm{GID}}$.

$$\begin{aligned} C_{2,\mathrm{GID}} &= \prod_{x\in I} \left(e\left(\mathrm{CSK}_{\rho(x),\mathrm{GID}}, C_{2,x}\right) \times e(H_1(\mathrm{GID})^{x_{\mathrm{GID}}}, \right. \\ &\quad \left. C_{3,x} g^{C_{6,x}}) e\left(\mathrm{CSK}'_{\rho(x),\mathrm{GID}}, C_{4,x}\right)\right)^{c_x} \\ &= \prod_{x\in I} \left(e\left(g^{x_{\mathrm{GID}} \alpha_{\delta(x)}} H_1(\mathrm{GID})^{x_{\mathrm{GID}} y_{\delta(x)}} H_3(\rho(x))^{t_{\rho(x)}}, g^{-r_x}\right)\right. \\ &\quad \left. \times e(H_1(\mathrm{GID})^{x_{\mathrm{GID}}}, g^{y_{\delta(x)} r_x} g^{\omega_x}) e\left(g^{t_{\rho(x)}}, H_3(\rho(x))^{r_x}\right)\right)^{c_x} \\ &= \prod_{x\in I} \left(e(\mathrm{g}, \mathrm{g})^{x_{\mathrm{GID}} \alpha_{\delta(x)}(-r_x) c_x}\right) (e(H_1(\mathrm{GID}), g)^{x_{\mathrm{GID}} \omega_x c_x}) \\ &= e(g, g)^{\sum_{x\in I} x_{\mathrm{GID}} \alpha_{\delta(x)}(-r_x) c_x}. \end{aligned} \tag{2}$$

MDU computes formula (5) to acquire $e(g,g)^s$.

$$C_{1,GID}(C_{2,GID})^{1/x_{\mathrm{GID}}} = e(g,g)^{s+\sum_{x\in I}\alpha_{\delta(x)}r_x c_x} \times \left(e(g,g)^{\sum_{x\in I}x_{GID}\alpha_{\delta(x)}(-r_x)c_x}\right)^{1/x_{\mathrm{GID}}} = e(g,g)^{s+\sum_{x\in I}\alpha_{\delta(x)}r_x c_x}\left(e(g,g)^{\sum_{x\in I}\alpha_{\delta(x)}(-r_x)c_x}\right) = e(g,g)^s. \tag{3}$$

It recovers $Y = C_0/e(g,g)^s$ and computes $\mathrm{key} = H_4(Y)$, $m' = \mathrm{SE.Dec}_{\mathrm{key}}(M_{\mathrm{SE}})$. MDU checks if there is a redundancy 0^ω appended after m'; if it is, $m' = m\|0^\omega$. Then, it checks if $H_1(m') = \mathrm{Ver}$ (Ver can be obtained in blockchain); if it is, the message m can be obtained by truncating ω-bit 0 string. Otherwise, CSP is dishonest to return an incorrect semidecrypted ciphertext and algorithm outputs $\perp$.

5.2.2. Correctness of Decryption of Reencryption Ciphertext $ReKey_{(S,GID)\longrightarrow(M',\rho')}$. First, compute C_4 and C_5 separately.

$$\begin{aligned} C_4 &= \prod_{x\in I}\left(e\left(C_{1,x}g^{C_{5,x}}, rk_3\right)\right)^{c_x} = \prod_{x\in I}\left(e\left(g^{\alpha_{\delta(x)}r_x}g^{\lambda_x}, g^{H_2(\beta)}\right)\right)^{c_x} \\ &= \prod_{x\in I}\left(e(g,g)^{\lambda_x H_2(\beta)c_x}e(g,g)^{\alpha_{\delta(x)}r_x H_2(\beta)c_x}\right) \\ &= e(g,g)^{sH_2(\beta)+\sum_{x\in I}\alpha_{\delta(x)}r_x H_2(\beta)c_x}, \end{aligned} \tag{4}$$

$$\begin{aligned} C_5 &= \prod_{x\in I}\left(e(rk_1, C_{2,x})e\left(rk_4, C_{3,x}g^{C_{6,x}}\right)e(rk_2, C_{4,x})\right)^{c_x} \\ &= \prod_{x\in I}\left(e(\mathrm{g},\mathrm{g})^{\alpha_{\delta(x)}H_2(\beta)(-r_x)}e\left(H_1(\mathrm{GID}),\mathrm{g}\right)^{H_2(\beta)\omega_x}\right)^{c_x} \\ &= \prod_{x\in I}e(\mathrm{g},\mathrm{g})^{\alpha_{\delta(x)}H_2(\beta)(-r_x)c_x}e\left(H_1(\mathrm{GID}),\mathrm{g}\right)^{H_2(\beta)\omega_x c_x} \\ &= e(g,g)^{\sum_{x\in I}\alpha_{\delta(x)}H_2(\beta)(-r_x)c_x}. \end{aligned} \tag{5}$$

Then, compute C_6.

$$C_6 = C_4 \bullet C_5 = e(g,g)^{sH_2(\beta)} \tag{6}$$

The CSP checks whether S' matches (M',ρ') in $C^R_{(M',\rho')}$; if S' does not match (M',ρ'), it aborts. Otherwise, CSP performs the following steps: CSP let $I' = \{x : \rho'(x) \in S'\} \subseteq \{1,2,\cdots,l'\}$ and computes $\{c'_x \in Z_p^*\}$ to let $\sum_{x\in l'}c'_x M'_x = (1,0,..,0)$. CSP computes formula (6) to acquire $C_{1,\mathrm{GID}'}$.

$$\begin{aligned} C_{1,\mathrm{GID}'} &= \prod_{x\in I'}e\left(C'_{1,x}, g\right)^{c_x'} = \prod_{x\in I'}e\left(g^{\lambda_x'}g^{\alpha_{\delta'(x)}r'_x}, g\right)^{c'_x} \\ &= \prod_{x\in I'}e(g,g)^{\lambda_x' c_x'}e(g,g)^{\alpha_{\delta'(x)}r'_x c'_x} = e(g,g)^{s'+\sum_{x\in I'}\alpha_{\delta'(x)}r'_x c'_x}. \end{aligned} \tag{7}$$

CSP then computes formula (7) to acquire $C_{2,\mathrm{GID}'}$.

$$\begin{aligned} C_{2,\mathrm{GID}'} &= \prod_{x\in I'}\left(e\left(CSK_{\rho'(x),\mathrm{GID}'}, C'_{2,x}\right)e\left(H_1\left(\mathrm{GID}'\right)^{x_{\mathrm{GID}'}}, C'_{3,x}\right)\right. \\ &\quad \left.\times e\left(\mathrm{CSK}'_{\rho'(x),\mathrm{GID}}, C'_{4,x}\right)\right)^{c'_x} \\ &= \prod_{x\in I'}e\left(g^{x_{\mathrm{GID}'}\alpha_{\eta'}}H_1\left(\mathrm{GID}'\right)^{x_{\mathrm{GID}'}y_{\eta'}}H_3(\rho(x))^{t_{\rho(x)}}, g^{-r_x'}\right) \\ &\quad \times e\left(H_1\left(\mathrm{GID}'\right)^{x_{\mathrm{GID}'}}, g^{y_{\delta'(x)}r'_x}g^{\omega_x'}\right) \\ &\quad \cdot e\left(g^{t_{\rho(x)}}, H_3(\rho(x))^{r_x'}\right)\Big)^{c'_x} = e(g,g)^{\sum_{x\in I'}x_{\mathrm{GID}'}\alpha_{\eta'}(-r'_x)c'_x}. \end{aligned} \tag{8}$$

MDU computes formula (8) to acquire β.

$$\begin{aligned} \beta &= = \frac{\beta\bullet e(g,g)^{s'}}{e(g,g)^{s'+\sum_{x\in I'}\alpha_{\delta'(x)}r'_x c'_x}\left(e(g,g)^{\sum_{x\in I'}x_{GID}'\alpha_{\eta'}(-r'_x)c'_x}\right)^{1/x_{\mathrm{GID}'}}} \\ &= \frac{\beta\bullet e(g,g)^{s'}}{e(g,g)^{s'}}. \end{aligned} \tag{9}$$

Then MDU can recover $Y = C_0/(C_6)^{1/H_2(\beta)}$. MDU computes $\mathrm{key} = H_4(Y)$ and $m' = \mathrm{SE.Dec}_{\mathrm{key}}(M_{\mathrm{SE}})$. The algorithm then checks if there is a redundancy 0^ω appended after m'; if it is, $m' = m\|0^\omega$. Then it checks if $H_1(m') = \mathrm{Ver}$ (Ver can be obtained in blockchain), if it is, the message m can be acquired by omitting ω-bit 0 string. If not, CSP is dishonest to return an incorrect semidecrypted ciphertext and the algorithm outputs $\perp$.

5.3. Security Analysis

Theorem 2. *If the q-DBPBDHE2 assumption holds, then the proposed scheme is statically secure under ROM.*

The proof of Theorem 2 is based on the establishment of Lemmas 3 and 4. Therefore, the proofs of Lemmas 3 and 4 will be given as follows, respectively.

Lemma 3. *If the q-DBPBDHE2 assumption holds, then the RW scheme proposed in [32] is statically secure under ROM.*

Proof. Lemma 3 is proved in [32], so Lemma 3 is valid. □

Lemma 4. *If the RW scheme constructed in [32] is statically secure under ROM, then the proposed scheme is statically secure under ROM.*

Proof. Assuming there is a PPT adversary *Adv* that can break the proposed scheme with a non-negligible advantage ε, then there must exist a simulator B who can break the RW scheme with advantage ε. Among them, B can break the proposed scheme with Adv and the challenger C in RW. The relevant phases are as follows.

(1) Initialization phase: *C* sends the public parameter *PP* in the RW scheme to the simulator *B*. *B* performs $GlobalSetup(\lambda, A) \longrightarrow (PP, R)$ algorithm and sends the uploaded public parameter $PP = \{p, G, G_T, g, e, H_1, H_2, H_3, H_4, SE, A, A_\eta\}$ to *Adv*.

(2) Query phase: *Adv* does the following query to *B*.

□

Query 1: *Adv* chooses some N_θ, *Adv* asks *B* for the corresponding public key NPK_θ, *B* then sends NPK_θ to *C*. *C* performs $AuthoritySetup(GP, \theta) \longrightarrow (UPK_\theta, USK_\theta)$ in [32] to generate $UPK_\theta = \{e(g, g)^{\alpha_\theta}, g^{y_\theta}\}$ and sends it to *B*. After that, *B* performs $NodeSetup(PP, \theta) \longrightarrow (NPK_\theta, NSK_\theta)$ to generate $UPK_\theta = \{e(g, g)^{\alpha_\theta}, g^{y_\theta}\}$, then *B* updates public key and sends them to *Adv*.

Query 2: *Adv* selects some legitimate users $\{GID_i\}_{i=1}^m$ and asks for their key pairs $\{UPK_i, USK_i\}_{i=1}^m$. *B* performs $UserKeyGen(PP, GID) \longrightarrow (UPK, USK)$ to generate the corresponding $UPK_i = (g^{x_{GID_i}}, H_1(GID)^{x_{GID_i}})$ and $USK_i = x_{GID_i}$, then sends (UPK_i, USK_i) to *Adv*.

Query 3: *Adv* asks *B* for the transfer key: *B* first forwards the query to *C* to get the result output by scheme [32]. Then *B* performs $TKGen(PP, GID, S, NSK_\theta, UPK) \longrightarrow (TK_{S,GID})$ to generate transfer key $TK_{S,GID}$ and return it to *Adv*. During the inquiry process, *B* calculates the transfer key corresponding to $\{S_i, GID\}_{i=1}^n$ in two cases, specifically:

Case 1. For $1 \le i \le m$, attribute $j \in S_i$, *B* randomly selects $t_j \in Z_p^*$ and computes $CSK_{j,GID_i} = (g^{\alpha_\theta} H_1(GID_i)^{y_\theta} H_3(j)^{t_j})^{x_{GID_i}} = g^{x_{GID_i}\alpha_\theta} H_1(GID_i)^{x_{GID_i}y_\theta} H_3(j)^{t_j x_{GID_i}}$, $CSK'_{j,GID} = H_3(j)^{t_j x_{GID_i}}$. *B* then acquires the transfer key $TK_{S_i,GID_i} = (CSK_{j,GID_i}, CSK'_{j,GID})_{j \in S_i}$.

Case 2. For $m \le i \le n$, attribute $j \in S_i$, *B* randomly selects $t_j \in Z_p^*$, $g_j \in G$, computes $CSK_{j,GID_i} = g_j H_3(j)^{t_j}$ and $CSK'_{j,GID} = H_3(j)^{t_j}$. Since $g^{\alpha_\theta} H_1(GID_i)^{y_\theta}$ is in *G*, there must be an unknown $x_{GID_i} \in Z_p^*$ that can make $g_j = (g^{\alpha_\theta} H_1(GID_i)^{y_\theta})^{x_{GID_i}} = g^{x_{GID_i}\alpha_\theta} H_1(GID_i)^{x_{GID_i}y_\theta}$. Thus, the corresponding transfer key can be obtained as $CSK_{j,GID_i} = g_j H_3(j)^{t_j} = g^{x_{GID_i}\alpha_\theta} H_1(GID_i)^{x_{GID_i}y_\theta} H_3(j)^{t_j}$, $CSK'_{j,GID} = H_3(j)^{t_j}$. *B* then sends the transfer key $TK_{S_i,GID_i} = (CSK_{j,GID_i}, CSK'_{j,GID})_{j \in S_i}$ to *Adv*.

Table 2: System function comparison of different schemes.

Scheme	[28]	[24]	[23]	[20]	[19]	Ours
Support ciphertext conversion	×	√	√	√	√	√
Collusion resistant	√	√	√	√	√	√
Decryption outsourcing	√	√	√	×	×	√
No KGC	√	×	×	×	×	√
User revocation	√	×	×	√	√	√
Decryption verification	×	√	×	×	×	√

Query 4: *Adv* asks *B* for re-encryption key: *B* first forwards the query to *C* to get the result output by scheme [32]. Then, *Adv* sends users' $GID_i (1 \le i \le n)$, users' attribute sets $S_i \in A$ and a new access policy (M', ρ') to *B*. *B* performs $ReKeyGen(PP, TK_{S,GID}, USK, NPK_\eta, S, (M', \rho')) \longrightarrow ReKey_{(S,GID)\longrightarrow(M',\rho')}$ to generate re-encryption key $ReKey_{(S,GID)\longrightarrow(M',\rho')}$. Then *B* returns $ReKey_{(S,GID)\longrightarrow(M',\rho')}$ to *Adv*. $TK_{S,GID}$ is generated by $TKGen()$ algorithm.

(3) Challenge Phase: *Adv* submits two equal-length plaintexts $M_0{}^*$, $M_1{}^*$, and an access policy (M^*, ρ^*) to *C*. *C* randomly selects $b \in \{0, 1\}$, executes $Enc()$ algorithm to calculate challenge ciphertext and return it to *Adv*. The limitation is that the $\{GID_i\}$ $(1 \le i \le m)$ of the MDUs who asked for the private key during the query, their attribute set S_i does not satisfy (M^*, ρ^*).

(4) Guess Phase: *Adv* outputs guess $b' \in \{0, 1\}$ on b. At this time, *B* gives different guess values b' according to the different guesses of *Adv*.

In the above game, *Adv* can consider that *B* is the challenger *C* in scheme [32] The transfer key sent by *C* corresponds to the user private key generated by *UserKeyGen* algorithm in the scheme [32]. At the same time, *B* can determine the symmetric key *key* in the proposed scheme according to the value of b'. Mainly, the *key* corresponds to the message *M* input by the *Encrypt* algorithm in the scheme [32]. Since the RW scheme is statically secure, the proposed scheme is also statically secure. Lemma 4 is proved.

In summary, from the proofs of Lemmas 3 and 4, it can be proved that Theorem 2 is also valid. Therefore, the proposed scheme is statically secure under ROM.

6. Performance Analysis

In this section, the function and efficiency of the proposed scheme and other related schemes [19, 20, 23, 24, 28] are evaluated and compared. We compared the aforementioned 5 schemes with our scheme in terms of system functions. The specific comparison is shown in Tables 2 and 3.

Table 3: System efficiency comparison of different schemes.

Scheme	[28]	[24]	[23]	[20]	[19]	Ours
Encrypt	$(7l+4)E+P$	$(6l+2)E$	$(7l+2)E$	$(5l+1)E$	$(6l+1)E$	$(6l+1)E$
Original ciphertext decrypt	E	$(3\|I\|+1)P+\|I\|E$	E	$(3\|I\|+2)P+E$	$2\|I\|P+2\|I\|E$	E
Reencrypt key	−	$2\|S\|+5E$	$\|S\|+l+6E$	$(2\|S\|+2l+3)E$	$4\|S\|E$	$(2\|S\|+8l+8)E$
Reencrypt ciphertext decrypt	−	$4P+2E$	$2E$	$3\|I\|P+2P+E$	$2\|I\|P+2\|I\|E$	$2E$
Total	$(7l+5)E+P$	$(6l+\|I\|+9)E+(3\|\ I\|+5)P+2\|S\|$	$(7l+11)E+\|\ S\|+l$	$(5l+2\|S\|+6)E+\ (6\|I\|+4)P$	$4\|I\|P+(\ 4\|S\|+4\|I\|+6l+1)E$	$(2\|S\|\ +14l+12)E$

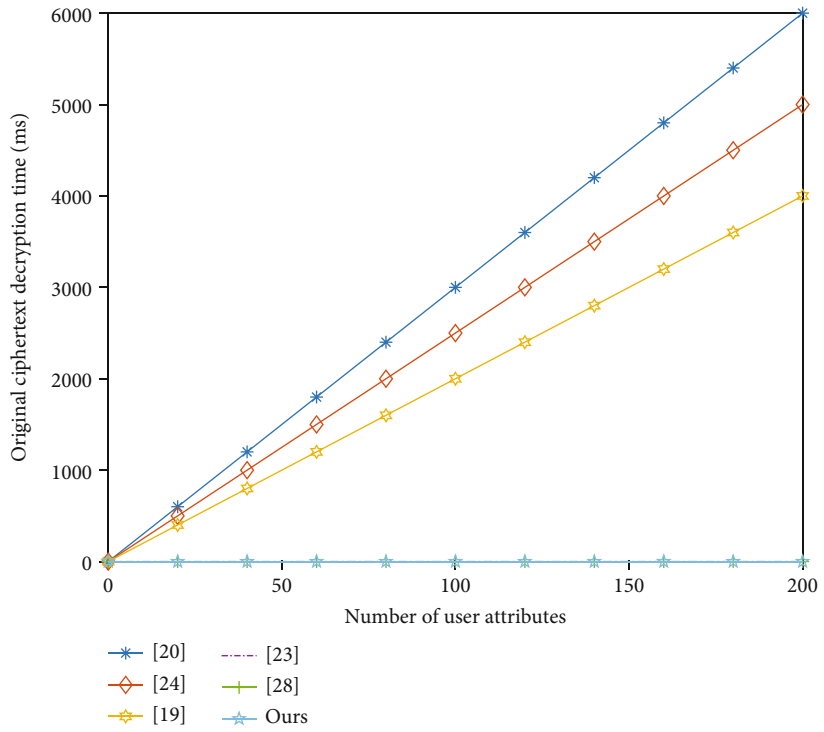

Figure 3: User original ciphertext decryption time.

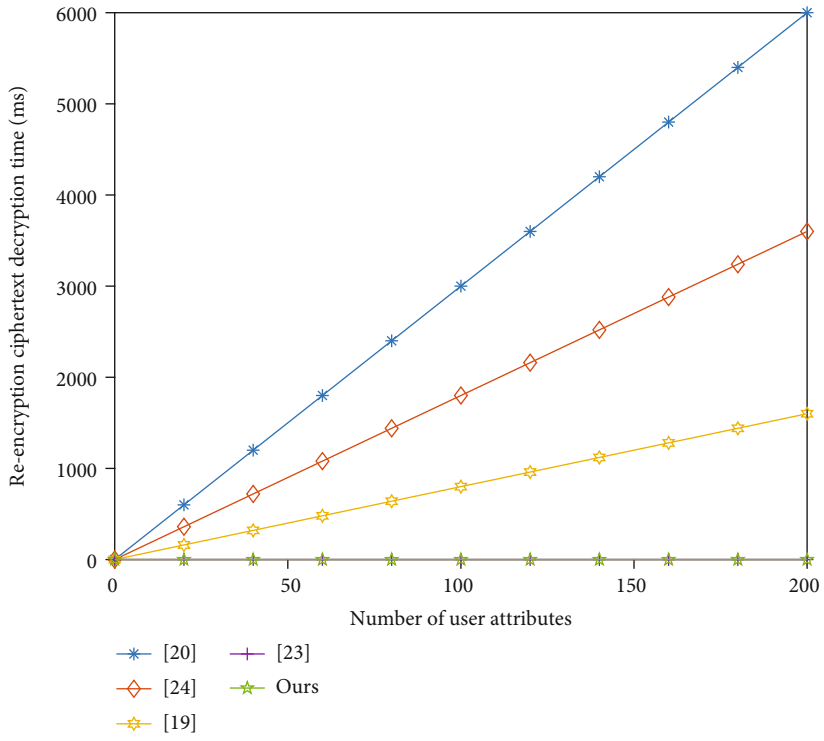

Figure 4: User reencrypted ciphertext decryption time.

Figures 3–5 shows the operating time of different algorithms. We use Java language in the Windows10 environment, and Java Pairing Based Cryptography (JPBC) is used as the cryptography library. G and G_T are selected from the elliptic curve $y2 = x3 + x$. $e(g, g)$ is a symmetric bilinear map. The length of the elements in G and G_T is 1024 bits. The AES encryption algorithm is used as a symmetric-key encryption algorithm. The hardware is AMD R5 4600u CPU, 2.1 GHz frequency, and 8GB memory.

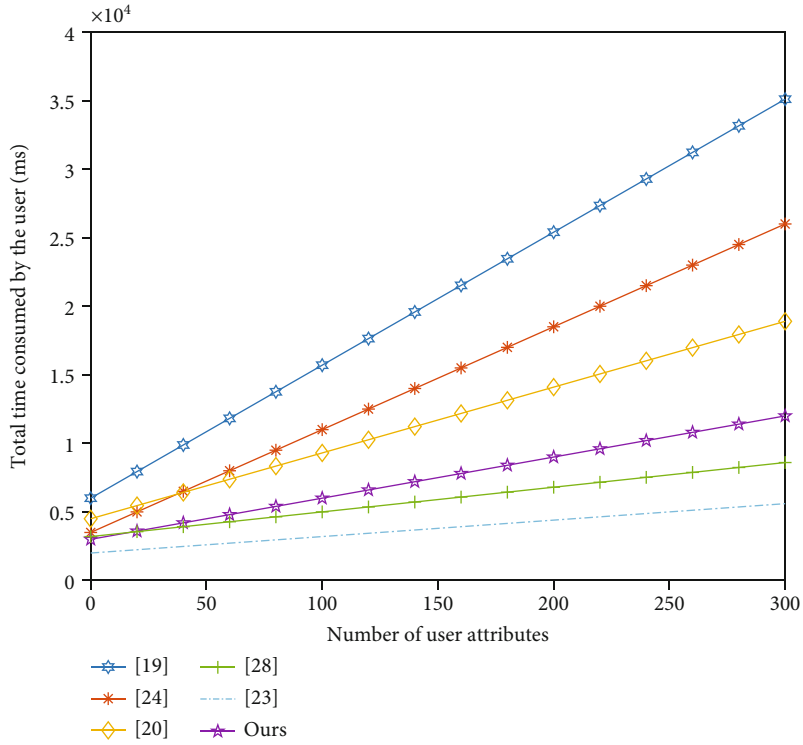

Figure 5: Total computational cost.

Table 2 shows the function comparison; schemes [19, 20, 23, 24] support ciphertext reencryption and collision resistance. Compared with the 4 schemes, the proposed scheme has the most comprehensive functions. Though scheme [19, 20] support ciphertext reencryption, they do not support outsourced decryption. Thus, MDU will face huge decryption costs. Although schemes [19, 20, 28] realize user revocation, the keys of unrevoked users need to be updated when the revocation occurs, which brings additional computational overhead. The user private keys in schemes [19–24] are all generated by KGC; once KGC is compromised, it will cause serious information security incidents. In fact, there exists no fully trusted KGC in the world. The proposed scheme does not need a KGC, and the user private keys are generated by users. Besides, consortium blockchain is introduced to ensure medical data will not be tampered with; consensus nodes work together to ensure blockchain operation. Schemes [20, 24] cannot realize user revocation. Even if a user leaves the system, his private key can still decrypt the ciphertext in the system, which may lead to abuse of key authority. Although schemes [20, 28] support decryption outsourcing, they lack decryption verification. This makes data users cannot distinguish whether the CSP has tampered with the result of outsourced decryption. In addition, the scheme [23] only realizes the reencryption from ABE to IBE, not the reencryption from ABE to ABE. ABE can realize "one-to-many" encryption, while IBE can only realize "one-to-one" encryption. In summary, the proposed scheme is practical in functions and is suitable for cloud storage environments.

Table 3 shows the system efficiency comparison of different schemes. For simplicity, $|S|$ denotes the number of user's attributes, l denotes the number of rows in M, $|I|$ denotes the number of rows used for decryption in M, P denotes the bilinear pairing operation in G, and P denotes the exponential operation in G. Since the calculation cost of the schemes is related to l, $|I|$, $|S|$, the variables are unified for the convenience of discussion. Let the user attribute be a subset of the policy attribute; the user attributes always satisfy the access policy so that $|I|$ is consistent with $|S|$. Thus, the computational cost of the solution is related to l and $|S|$.

It can be seen from the results given in Table 2 and Figures 3–5, schemes [23, 28] have a lower computational cost. The main reason is that in scheme [23], the ciphertext after reencryption is in IBE form, not ABE form, which makes its reencryption key cost is low. Scheme [28] does not support reencryption, so it has a low computational cost. Meanwhile, although schemes [20, 24] share the computational cost of the reencryption key to the trusted center, compared with the proposed scheme, the above two schemes have higher computational costs in the decryption stage. The computational cost of scheme [19] is the highest in all phases. Finally, from the perspective of total computational cost, the users in the proposed scheme have a lower computational overhead than users in schemes [19, 20, 24]. Therefore, the proposed scheme is efficient in computing.

7. Conclusion

This paper proposed a medical data sharing scheme supporting ciphertext reencryption. The proposed scheme can employ attribute-based proxy reencryption technology to allow the cloud to reencryption ciphertext. The reencryption key was provided by a user with decryption authority and

the cloud cannot obtain the plaintext. It can effectively reduce the bandwidth and time cost caused by the round-trip transmission of ciphertext. Simultaneously, the proposed scheme makes use of a consortium blockchain to store public keys and public parameters and update the revocation list. The data stored on the blockchain is secure and cannot be tampered with. Consensus nodes jointly maintain the consortium blockchain, which can prevent security and privacy issues caused by KGC's overcentralization. Ultimately, based on the q-DBPBDHE2 difficult problem assumption, the proposed scheme is proved to be statically secure. Hence, the proposed scheme is practical for secure access control and sharing of medical data in the cloud storage environment.

Data Availability

All data, models, and codes generated or used during the study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61972094) and the Foundation of State Key Laboratory of Information Security (Grant No. 2021-ZD-02).

References

[1] T. Wood, K. Ramakrishnan, P. Shenoy, J. Merwe, and J. Hwang, "CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines," *IEEE-ACM Transactions on Networking*, vol. 23, no. 5, pp. 1568–1583, 2015.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[3] B. Singh, S. Dhawan, A. Arora, and A. Patail, "A view of cloud computing," *International Journal of Computers & Technology*, vol. 4, no. 1, pp. 50–58, 2013.

[4] S. Xu, J. Ning, Y. Zhang, G. Xu, X. Huang, and R. Deng, "A secure EMR sharing system with tamper resistance and expressive access control," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1–223, 2021.

[5] A. Abbas and S. Khan, "A review on the state-of-the-art privacy-preserving approaches in the e-health clouds," *IEEE Journal of Biomedical Health Informatics*, vol. 18, no. 4, pp. 1431–1441, 2014.

[6] G. Anthes, "Security in the cloud," *Communications of the ACM*, vol. 53, no. 11, pp. 16–18, 2010.

[7] C. Chu, W. Zhu, J. Han, J. Liu, J. Xu, and J. Zhou, "Security concerns in popular cloud storage services," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 50–57, 2013.

[8] N. Kaaniche and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, vol. 111, pp. 120–141, 2017.

[9] A. Sari, "A review of anomaly detection systems in cloud networks and survey of cloud security measures in cloud storage applications," *Journal of Information Security*, vol. 6, no. 2, pp. 142–154, 2015.

[10] A. Almutairi, M. Sarfraz, S. Basalamah, A. Walid, and A. Ghafoor, "A distributed access control architecture for cloud computing," *IEEE Software*, vol. 29, no. 2, pp. 36–44, 2012.

[11] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 1998)*, pp. 127–144, Nyberg, Kaisa, 1998.

[12] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, https://bitcoin.org/bitcoin.pdf.

[13] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques (EUROCRYPT 2005)*, pp. 457–473, Aarhus, Denmark, 2005.

[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security (CCS 2006)*, pp. 89–98, Alexandria, VA, USA, 2006.

[15] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP 2007)*, pp. 321–334, Berkeley, CA, USA, 2007.

[16] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, pp. 53–70, Springer, Berlin Heidelberg, 2008.

[17] X. Liang, Z. Cao, L. Huang, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS 2009)*, pp. 276–286, Sydney, Australia, 2009.

[18] K. Liang, L. Fang, W. Susilo, and D. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCOS 2013)*, pp. 552–559, Xian, China, 2013.

[19] Y. Yang, H. Zhu, H. Lu, J. Weng, and R. Choo, "Cloud based data sharing with fine-grained proxy re-encryption," *Pervasive and Mobile Computering*, vol. 28, no. 2, pp. 122–134, 2016.

[20] X. Feng, C. Li, D. Li, Y. Fang, and Q. Shen, "Fully secure hidden ciphertext-policy attribute-based proxy re-encryption," in *International Conference on Information and Communications Security (ICIC 2017)*, pp. 192–204, Beijing, China, 2017.

[21] H. Ma, R. Zhang, Z. Wan, L. Yao, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 6, pp. 679–692, 2017.

[22] S. Xu, Y. Li, and R. Deng, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, vol. 99, no. 6, pp. 190–207, 2019.

[23] H. Deng, Z. Qin, Q. Wu, Z. Guan, and Y. Zhou, "Flexible attribute-based proxy re-encryption for efficient data sharing," *Information Sciences*, vol. 511, no. 11, pp. 94–113, 2020.

[24] A. Paul, S. Selvi, and P. Rangan, "Efficient attribute-based proxy re-encryption with constant size ciphertexts," in

International Conference on Cryptology in India (INDOCRYPT), pp. 644–665, Bangalore, India, 2020.

[25] S. Xu, J. Ning, X. Huang, Y. Li, and G. Xu, "Untouchable once revoking: a practical and secure dynamic EHR sharing system via cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1–240, 2021.

[26] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *International Conference on Advanced Communication Technology (ICACT 2017)*, pp. 464–467, Pyeong Chang, Republic of Korea, 2017.

[27] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *IEEE International Conference on Web Services (ICWS 2017)*, pp. 468–475, Honolulu, HI, USA, 2017.

[28] S. Hu, C. Cai, Q. Wang, C. Wang, and D. Ye, "Augmenting encrypted search: a decentralized service realization with enforced execution," *IEEE Transactions of Dependable and Secure Computing*, vol. 16, no. 6, pp. 2569–2581, 2019.

[29] S. Xu, J. Ning, Y. Li, Y. Zhang, G. Xu, and X. Huang, "Match in my way: fine-grained bilateral access control for secure cloud-fog computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 2193–2207, 2020.

[30] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, "Secure, efficient and revocable multi-authority access control system in cloud storage," *Computers & Security*, vol. 59, no. 5, pp. 45–59, 2016.

[31] M. Bellare, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security (CCS 1993)*, pp. 62–73, Fairfax, USA, 1993.

[32] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *International Conference on Financial Cryptography and Data Security (FC 2015)*, pp. 315–332, Sanjuan, Rico, 2015.