Hindawi
Security and Communication Networks
Volume 2018, Article ID 6174830, 13 pages
https://doi.org/10.1155/2018/6174830

Research Article

A Cheating Detectable Privacy-Preserving Data Sharing Scheme for Cloud Computing

Xin Wang,[1,2,3] Bo Yang,[1,3] Zhe Xia,[4] Yanqi Zhao,[1] and Huifang Yu[5]

[1]*School of Computer Science, Shaanxi Normal University, Xi'an 710119, China*
[2]*College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China*
[3]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[4]*School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China*
[5]*School of Communication and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China*

Correspondence should be addressed to Bo Yang; byang@snnu.edu.cn

Received 6 February 2018; Revised 2 July 2018; Accepted 7 August 2018; Published 16 October 2018

Academic Editor: Jun Zhou

Cloud computing provides a new, attractive paradigm for the effective sharing of storage and computing resources among global consumers. More and more enterprises have begun to enter the field of cloud computing and storing data in the cloud to facilitate the sharing data among users. However, in many cases, users may be concerned about data privacy, trust, and integrity. It is challenging to provide data sharing services without sacrificing these security requirements. In this paper, a data sharing scheme of reliable, secure, and privacy protection based on general access structure is introduced. The proposed scheme is not only effective and flexible, but also is capable of protecting privacy for the cloud owner, supporting data sharing under supervision, enabling accountability of users' decryption keys, and identifying cheaters if some users behave dishonestly. Security analysis and efficiency analysis demonstrate that our proposed scheme has better performance in computational costs compared with most related works. The scheme is versatile to be used in various environments. For example, it is particularly suitable to be employed to protect personal health data and medical diagnostic data in information medical environment.

1. Introduction

At present, new technologies and new industries emerge in an endless stream based on big data and cloud computing. Data production, management, and emerging business models based on big data are springing up. In recent years, with the transformation of large number of businesses into digitalization and informatization, mass data is constantly being manufactured and consumed, which promotes the rapid development of big data technology in research, development, and application. In the context of this industry, the quantity and quality of data become extremely important. Cloud computing provides a new and appealing paradigm to share the resources of storage and computation efficiently among the global consumers. Compared with the traditional data storage methods, cloud users can more conveniently access the data without considering the arrangement of the hardware or infrastructure by data storing and sharing on cloud. However, the cloud presents the value attraction for its huge functionality and convenience, and it brings a lot of new challenge. One reason is that the data owner has lost the physical control on the data when he stored the data on the cloud, and meanwhile, the cloud server is faced to the public. Hence, the owner's data may be subjected to various kinds of threats and malicious attacks. For instance, the data's confidentiality requirement may be disobeyed by some clouds for financial purposes, or they may even sell their business competitors confidential information. Thus, although cloud computing is very attractive to enterprises and consumers by economically sharing massive data among the users, it may fail to guarantee data storage security and privacy to individual of the data owner. Furthermore, in some uses, after the data possessor has put out his encrypted data to the cloud, he may yet wish to keep the

data's some controls, for example, update the data or revoke the access rights for some other users [1]. In consequence, many recent works have devoted to guarantee security and privacy using remotely storing the shared data and in the meantime assuring the desirable security characteristics. In 2010, the first scheme achieving secure data access control with provable security in cloud storage has been proposed by Yu et al. [2], using key-policy attribute-based encryption and symmetric encryption (KP-ABE) [3]. The scheme can reach fine-grained data access by combing KP-ABE with proxy reencryption (PRE) and lazy reencryption. The scheme's performance still needs to be improved, though a part of the private key update calculation can be put out to the cloud. Dong et al. subsequently give a scheme [4] employing symmetric encryption with ciphertext-policy attribute-based encryption [5, 6] (CP-ABE). To sum up, regarding the employed technologies, KP-ABE or CP-ABE has either been used by these above schemes under the symmetric encryption to design data security secure access control. Some other design methods are based on hierarchical identity-based encryption (HIBE) [7], but they have appealed to less attention. These schemes have simply considered how to supply data privacy against the cloud, but they do not have the preservation of private information of owner and the possible dishonest behaviors by some authorized users when storing the data in cloud. For instance, some authorized users may provide fake share deliberately to cause decryption failure. In the personal health medical information surroundings, some extra security demands are required. Since in certain special circumstances, such as medical accidents, abnormal deaths, and traffic accidents, medical disputes claim should be considered. In these situations, while medical evidence is needed, electronic medical records of the patient should be able to be right decrypted by the authorized users. Assuming someone has had an accident or been killed in serious incidents. In case supposing that there are disputes, historic records of the users will be the crucial evidences and they need to be recovered for expert testimony. These existing schemes, however, focus on the confidentiality and privacy of the data itself and the cost of performance mainly, but they are not suitable for medical records scenarios' needs. To realize an effectual, scalable, and privacy-preserving data sharing service in the cloud, the following requirements should be consequently satisfied:

(1) the data owner can authorize who can access the data, and the authorized users should be able to get to shared data in the cloud under the constraints that are defined by the data owner;

(2) the cloud needs to be able to give support to dynamic requirements so that data owners can update the data file and add or revoke users;

(3) when data owner stores his information on the cloud without suitable protection, the data are readable by anyone since the cloud is publicly accessible. The personal private information, for example, their medical information and users' telephone number, consequently needs to be protected against the cloud, and it should not be made public;

(4) the data decryption operation should be carried out under mutual supervision, and the dishonest users need to be identified if they submit false shares. How to settle the above important issues has not been considered in cloud that computes yet, although a number of schemes have been proposed in the literature.

In this paper, we propose an effective, scalable and flexible privacy-preserving data sharing scheme to ensure semantic security and effective utilization of owner's data. In order to preserve the privacy of owner's sensitive information that may be unrelated to the data itself, Bloom filter hash function is used to hide data storage in cloud. The scheme employs secret sharing based on general access structure to preserve confidentiality of the owner's data against the cloud. In addition, Reed-Solomon (RS) encoding technique has been adopted to identify the dishonest user who presents a false share. In the proposed scheme, all authorized users are divided into groups by their identity when they register themselves with the protocol. Each data file is described by a set of group secrets, and for every group, such as U_k, has been assigned one master key α_k so that the data file can be successfully decrypted when these group secrets are correctly recovered. In addition, each cloud user is assigned into a group and every group secret is shared among the group users by utilizing secret sharing of general access structure.

To ensure correct sharing of the secret, this scheme defines a public-private key pair for each user. By combing secret sharing access structure and user's public key, the share or secret key is sent to user in every group. Therefore, each user could get a different key and he can check this share key by his private key $d_{ID_i \to j}$. The secret keys of group users are defined to reflect their group access privileges, so that all of the users should present their correctly shared keys without cheating. RS encoding technique is used to enable the identification of the cheater when he provided a fake share.

In the proposed protocol, the secret is shared among multiple participants, and only a quorum of these participants work together can recover the secret. In an ideal secret sharing, the secret share held by each participant has exactly the same size as the secret, and the size of all secret shares together is proportional to the number of the participants. Therefore, when considering the protocol as a whole, more information needs to be dealt with, but each participant's task remains the same. The benefit is that the secrecy and availability of the secret key are enhanced. Suppose the adversary wants to learn the secret key or destroy it. She needs to compromise multiple participants to achieve her objective instead of compromising a single one in the traditional protocols.

Compared with the existing schemes, our analysis shows that the proposed scheme provides the following benefits regarding both security and efficiency:

(1) The cloud server can assist search record by data file tag and it can learn nothing about owner's data in plaintext and owner's personal sensitive information.

(2) The user who can access the data file is authorized by the data owner, and he can verify the secret key sent by the owner.

(3) The dishonest cloud users who present fake decryption keys can be identified, so that the ciphertext can be

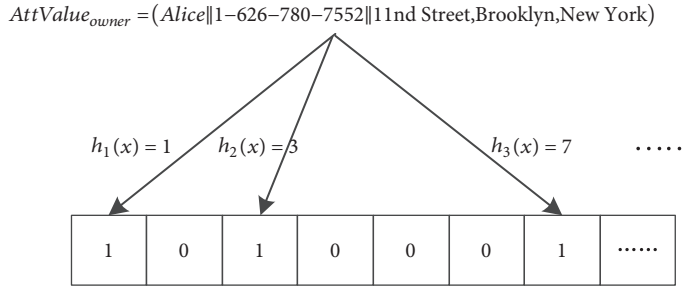

Figure 1: Bloom filter of personal privacy information.

safely and correctly decrypted under the supervision of these users.

The rest of this paper is organized as follows. The preliminaries are briefly described in Section 2. Section 3 discusses system models and security requirement. Our proposed scheme is introduced in Section 4, and its security is analyzed in Section 5. Efficiency analysis as well as its comparison with the related existing schemes is presented in Section 6. Finally, we present an example of the practical impact of our work and conclude this paper in Sections 7 and 8, respectively.

2. Preliminaries

2.1. Bilinear Maps. Let G_1 and G_2 be two multiplicative cyclic groups with prime order p and g be a generator of group G_1. Moreover, let $e : G_1 \times G_1 \longrightarrow G_2$ be the bilinear map that satisfies the following properties:

(1) Bilinearity: for all a and b there must be $e(g^a, g^b) = e(g, g)^{ab}$.

(2) Nondegeneracy: there must be $e(g, g) \neq 1$.

2.2. Secret Sharing Schemes. Secret sharing schemes (SSS) [8] are used to divide a secret among a number of parties. The value given to a party is called the share (of the secret) for that party. Every SSS realizes some access structure that defines the sets of parties who should be able to reconstruct the secret using their shares.

2.3. Access Structure and Monotone Span Programs

Definition 1 (access structure [9, 10]). Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotone for $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of nonempty subsets of $\{P_1, P_2, \cdots, P_n\}$; i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \setminus \{\varnothing\}$. The sets in $\mathbb{A}$ are called the authorized sets; otherwise, they are called the unauthorized sets.

In a linear secret sharing scheme [9], to realize an access structure $\mathbb{A}$, the dealer who possesses the secret y can distribute these shares of y to a number of parties such that y can be reconstructed by a linear combination of these shares of any authorized set. However, an unauthorized set can obtain no information about the secret y.

There is a close relationship between linear secret sharing scheme and a linear algebraic model called monotone span programs (MSP) [11]. It has been shown that the existence of a linear secret sharing scheme for some access structure is equivalent to the existence of a monotone span program for that access structure.

Definition 2 (monotone span program). Let $\mathcal{K}$ be a field and $\{x_1, \cdots, x_n\}$ be a set of variables. A monotone span program over $\mathcal{K}$ is a labeled matrix $\widehat{M}(M, \rho)$ where M is a matrix over $\mathcal{K}$ and ρ is a labeling of the rows of M by literals from $\{x_1, \cdots, x_n\}$ (every row is labeled by one literal).

A monotone span program accepts or rejects an input by the following criteria. For every input set γ of literals, let M_γ be the submatrix composing of those rows whose labels are in γ. The monotone span program $\widehat{M}$ accepts γ if and only if $\overrightarrow{1} \in span(M_\gamma)$.

2.4. Bloom Filter. Bloom filter (BF) [12] is a simple and effective random data storage structure. It is constructed by a set of hash functions $BF(x) = (h_1(x), \cdots, h_k(x))$ and it has two operations: $add(x)$ and $query(x)$, where x indicates Tag_{owner} in the proposed scheme. The $add(x)$ operation handles an element with multiple hash functions $h_1(\cdot), \cdots, h_k(\cdot)$, so that the element is uniformly mapped to a number, for example, $h_i(x) = y_i \in [1{:}m]$, and sets the y_i-th bit in the array to be one (the array is initialized to zeroes). The $query(x)$ operation repeats the same hashing procedure and then checks if the appropriate bits are set as 1. In 2012, the partially hidden access structure in ABE was proposed [13, 14]. In addition, the Bloom filter was employed to hide the value of the attribute in partially hidden access structures in [15]. In this proposed scheme, we use the Bloom filter to protect the privacy of the data owner as Figure 1.

In order to prevent the cloud server from learning information that may invade personal privacy, like name, mobile number, and home address, each attribute information is split into two parts: an attribute name and its value. The general attribute name is made public while the personal privacy information's specific attribute values are kept secret, when the data file is stored in the cloud server. For example, let the data owner's name be Alice, the phone number be 1-626-780-7552, and home address be 11ndStreet, Brooklyn, New York. Then, let the general attribute name of personal

information to be protected be $AttName_{owner} = (Name, Occupation, TelephoneNumber, Address)$, then the owner's specific values is $AttValue_{owner} = (Alice\|Lawyer\|1-626-780-7552\|11ndStreet, Brooklyn, New York)$. The data owner builds the data file label $Tag_{owner} = H(AttValue_{owner})$ and then constructs a Bloom filter $BF_{datafile} = BF(Tag_{owner})$ using Tag_{owner}. In Figure 1, let $x = H(AttValue_{owner})$.

To check whether a Tag is in set S that is stored in the cloud, we should firstly compute the values $h_1(Tag)$, $h_2(Tag), \cdots, h_k(Tag)$, and verify if each $h_i(Tag)$ is 1, where $i \in [1 \sim k]$. If the check fails, we can insure that Tag is not in S. Otherwise, we say that Tag is in S with a high probability, because the bloom filter has always a false positive rate. The false positive rate will be analyzed in detail in Section 6. The Bloom filter has an attractive feature of convenient query and concise space. When applying standard Bloom filter, there is a necessity to do k hash operations, where the time complexity to insert one element is $O(k)$. When determining whether an element is in the set, the k hash calculation is also needed. In addition, it needs the time complexity $O(k)$ to finish an element query. For a set with n elements, it just needs a bit array with size m, so the space complexity is $O(m)$. It is then very concise to use only m/n bits to save each element. The storage space of the traditional tree query algorithm and hash query algorithm is directly associated with the size of the element itself and the size of the set, while the Bloom filter query algorithm is independent of the number of the elements and it is simply connected the number of the vector's bits, where the mapping comes from the element to the vector.

2.5. Reed-Solomon Code. In coding theory, RS code could detect and correct a number of random information errors. McEliece and Sarwate [16] pointed out that Shamir's Secret Sharing Scheme (SSS) is closely related to the RS error correction. They observed that a list of shares of Shamir's (k, n) threshold SSS forms a codeword of RS code. Thus, if $k + 2t$ shares containing t invalid shares are provided in the reconstruction phase, the secret reconstruction algorithm can identify all t cheaters with certain probability. In addition, it is obvious, by using Lagrange interpolation, that a polynomial $f(x)$ of degree $k - 1$ is uniquely determined by $f(1), \cdots, f(n)$ if and only if $n \geq k + 2t$, where t is the number of the cheaters. In 2011, using a single keyed message authentication code, Obana designed an efficient (k, n) threshold SSS with unconditional security, which is capable of identifying up to t cheaters under the condition $(k - 1)/3 \geq t$ [17, 18].

3. System Model and Security Goals

3.1. System Model. In our system model, there are four participants: data owner, data consumers, cloud server, and public key generator (PKG). The data owner, e.g., the patient, stores his medical data in the cloud. In this way, he can outsource the data maintenance to the cloud. The data consumers download the data file shared by the data owner and decrypt it using their decryption keys. For the sake of simplicity, the data consumers are referred to as users in this paper. When decrypting data file, these users are collaborating with each other. The cloud server offers a high-quality service utilizing a large number of servers. It has considerable storage space and computation power. The data owner can interact with the cloud server dynamically to update or delete his data files. The public key generator maintains the public key infrastructure. It is a trusted third party with responsibility to deliver the decryption key safely from the data owner to the users. This framework for privacy-preserving data sharing in the cloud is shown as in Figure 2.

Note that, the communication channels between users and cloud server are secured under existing protocols in the system model.

3.2. Adversary Model. The adversary model defines malicious behaviors based on whether they intimidate the confidentiality of the cloud data. In contrast, the cloud server in our model is semitrusted (also known as honest-but-curious or passive. In other words, it will follow the protocol most of the time). The cloud server is assumed not to collude with the cloud user. Note that although the semitrusted adversary model is weaker than the malicious model, it is a realistic model that is widely used in similar protocols.

We have made it clear that the cloud server is semitrusted in our adversary model. This implies that the cloud server will not violate from the protocol but may try to learn more information that she is not authorized to access. Phishing attack is an important issue that needs to be considered in practice, but we will not address this issue since it is an actively attack. We will further consider this issue in our future works.

Hence, the following three types of attackers are considered: (1) data exposure: data owner's personal sensitive information might be leaked; (2) inner threats: some authorized users might present fake decryption key, causing failure when decrypting the data file; (3) outer threats: the channel that used to transmit the secret keys might be insecure.

Different from publicly verifiability [19], the user could verify by himself about the decryption key delivered by the owner, and this property is called secret verifiability.

3.3. Design Goals. With the purpose of secure data sharing and data access control in the cloud, our main goal is to minimize the leakage of the data owner's privacy information and prevent the malicious users from accessing the cloud data, including the deceptive users and collusive users. Then, the main design goals of our system can be summarized as follows.

(1) Personal Privacy Protection. We use bloom filter to design a secure mechanism so that the data owner and the cloud server can share data through the cloud. The operation only involves some hash operations, so the computational cost is very low.

(2) Data Access Control and Confidentiality. The proposed scheme employs secret sharing method to share data. The data owner has the authority to specify policy how these cloud users can access the data, and those unauthorized users cannot obtain the information of the data file.

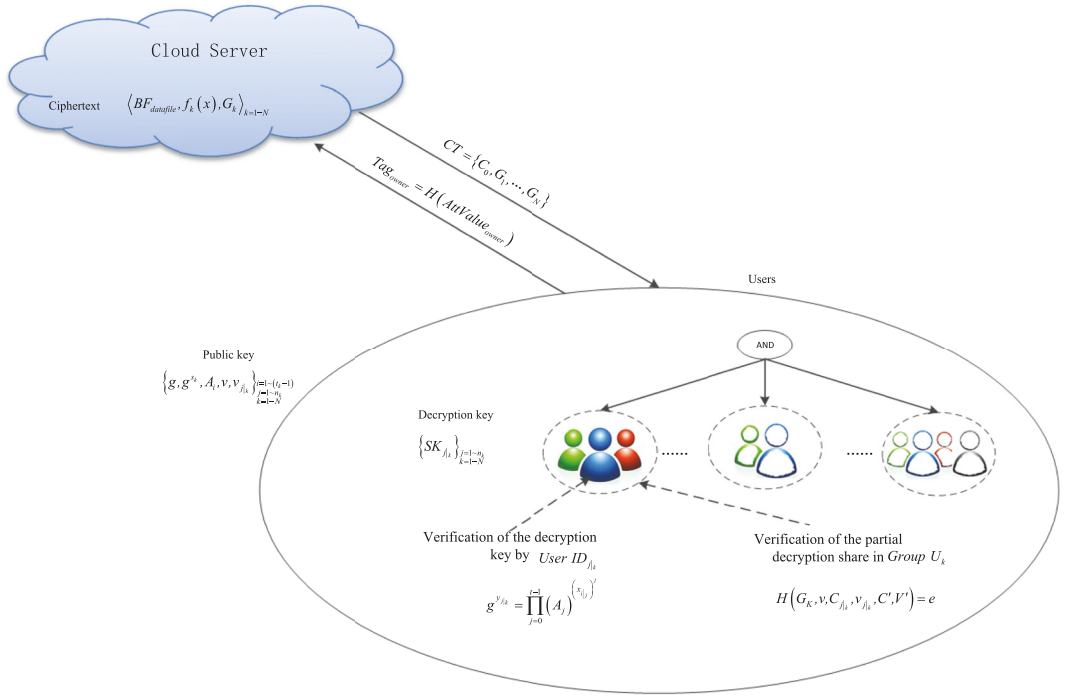

Figure 2: The framework for privacy-preserving data sharing service in the cloud.

(3) Secret Verifiability of the Decryption Key. In order to ensure secure communications over insecure channels, the decryption key can be verified by the user himself that it has been correctly delivered by the owner. This property is called secret verifiability, while in publicly verifiability, the key can be verified by anyone who is interesting to.

(4) Recognition of the Dishonest User. If some cloud users misbehave, they could be efficiently identified using the RS encoding method.

4. The Proposed Scheme

4.1. System Initialization. The public key generator chooses two groups G_1 and G_2 of prime order p, two independent generators $g, \hat{g}$, a bilinear map e:$G_1 \times G_1 \longrightarrow G_2$, and an injective function μ:GF(p) $\times \{1, \cdots, n\} \longrightarrow$ GF(q) (for example, $\mu(x, y) = (y-1)p + x$ [18]) and a collision-resistant hash function $H(\cdot)$.

(1) User registration: the user registers to the public key generator. He first randomly chooses $d_{ID} \in Z_p^*$ as the private key and then computes $h_{ID} = g^{d_{ID}}$ as the public key.

(2) Here, U denotes the set of users who want to share an owner's data file. Firstly, the public key generator takes a grouping function $\varphi(\cdot)$ and divides these users U into N different groups, such as doctors and nurses, relatives and friends, legal officers, and so on by user's identity, which are denoted as $U_1, \cdots, U_N$ that satisfy $U = U_1 \cup \cdots \cup U_N$. Suppose the user ID is partitioned into $U_{\varphi(ID)}$, where $\varphi(\cdot)$ is defined as $\varphi(ID)$:$ID \longmapsto \{1, \cdots, N\}$. If $\varphi(ID) = k$, where $k \in \{1, \cdots, N\}$, then the user group $U_{\varphi(ID)}$ is also denoted as U_k for short, namely, the group ID.

(3) PKG takes random exponents $a_k, \alpha_k \in Z_p$ for group U_k

(4) The system public key is published as

$$PK = \left\{g, q, \mu, H(\cdot), e(g,g)^{\alpha_1}, \cdots, e(g,g)^{\alpha_N}, g^{a_1}, \cdots, g^{a_N}\right\}. \quad (1)$$

$MSK = \{\alpha_1, \cdots, \alpha_N\}$ is denoted as the system master key.

4.2. Data File Generation (Data File Sharing)

(1) U_k is a partitioned group and the number of the users in the group U_k is l_k. The data owner chooses a linear secret sharing access structure and an exponent for every group. For group U_k, the data owner chooses a random secret sharing access structure (M_k, ρ_k), where ρ_k associates rows of the matrix M_k and $\rho_{k,j}$ corresponding to the jth row of M_k. Then it takes random exponents $\{s_k \mid k \in \{1, \cdots, N\}\}$. $\mathcal{M}$ is the data to be encrypted; the data file is published as

$$C = \left\{C_0 = \mathcal{M} \cdot e(g,g)^{\sum_k \alpha_k \cdot s_k}, C_1 = g^{s_1}, \cdots, C_N = g^{s_N}\right\}. \quad (2)$$

(2) In order to protect the private information of the data owner, the data owner computes the data file tag

Table 1: Data file format on the cloud server.

Document number	$BF_{datafile}$	$C_0 = \mathscr{M} \cdot e(g,g)^{\sum_k a_k \cdot s_k}, C_1 = g^{s_1}, \cdots, C_N = g^{s_N}$

$Tag_{owner} = H(AttValue_{owner})$, and then constructs a Bloom filter $BF_{datafile} = BF(Tag_{owner})$ by using Tag_{owner}.

(3) The data owner selects a unique ID for this data file and uploads the anonymous data file $\langle BF_{datafile}, (M_k, \rho_k), C_0, C_k\rangle_{k=1\sim N}$ to the cloud server.

Finally, each data file is stored on the cloud in the format as shown in Table 1.

4.3. Key Generation. User ID_i is a user of the universal user set, which is partitioned into group U_k by grouping function $\varphi(\cdot)$. To sign the user ID_i in group U_k, if user ID_i is the jth in group U_k, then it is denoted as $ID_{i\longrightarrow j|_{U_k}}$, for short $ID_{i\longrightarrow j}$.

(1) As mentioned earlier, the user ID_i takes random $d_{ID_i} \in Z_p^*$ as private key and computes $h_{ID_i} = g^{d_{ID_i}}$ as his public key.

(2) The data owner distributes the shared key to user ID_i in group U_k.

(a) (M_k, ρ_k) is the secret sharing access structure for group U_k, where $M_k = [m_{k_{ij}}]_{n_k \times l_k}$. Then the data owner chooses a random vector $\overrightarrow{v_k} = (s_k, y_2, \cdots, y_{l_k}) \in Z_p^n$ as secret vector, where s_k is the encryption exponent to share in the group U_k. Then the share vector is computed as $(\lambda_{k_1}, \cdots \lambda_{k_j} \cdots, \lambda_{k_{l_k}})^T = \overrightarrow{(M_k)_{\rho_{k,i}}} \cdot \overrightarrow{v_{\varphi(i)}}^T$, where $\overrightarrow{(M_k)_{\rho_{k,i}}}$ is the i-th row vector of the matrix M_k.

(b) The data owner chooses random $t_k \in Z_p^*$ and computes the share key $Sh_{ID_{i\longrightarrow j}} = h_{ID_i}^{a_k \cdot \lambda_{k_j}}, K_k' = g^{\alpha_k} \cdot g^{a_k \cdot t_k}, K_k'' = g^{t_k}$ and the share verification $\Omega_{ID_{i\longrightarrow j}} = \{(\sigma_{ID_{i\longrightarrow j}})_1 = \widehat{g}^{a_k \cdot y_1}, \cdots, (\sigma_{ID_{i\longrightarrow j}})_{l_k} = \widehat{g}^{a_k \cdot y_{l_k}}\}$ of the key for user $ID_{i\longrightarrow j}$, where $y_1 = s_k$.

(c) The data owner takes a random degree t polynomial $R_k(x) \in GF_q(x)$ and computes $R_k(\mu(Sh_{ID_{i\longrightarrow j}}, ID_{i\longrightarrow j}))$, denoted as $v_{R_j|_{U_k}}$, in short $v_{R_{ID_{i\longrightarrow j}}}$, with the injective function μ:GF$(p) \times \{1, \cdots, n\} \longrightarrow$ GF(q).

(d) $SK_{i\longrightarrow j} = \{Sh_{ID_{i\longrightarrow j}}, K_k', K_k'', \Omega_{ID_{i\longrightarrow j}}, v_{R_{ID_{i\longrightarrow j}}}\}$ is encrypted by the user's public key, so that only the user can decrypt it by his private key. In other words, the data owner sends the share key $SK_{i\longrightarrow j}$ safely to the user $ID_{i\longrightarrow j}$ by the public key infrastructure.

(3) The user receives the $SK_{i\longrightarrow j}$ and computes the decryption key by his private key d_{ID_i} as follows.

(a) The user first verifies $e(\prod_{j=1}^{l_k}(\sigma_{ID_{i\longrightarrow j}})_j^{m_{k_{ij}}}, g) = e(\widehat{g}, (Sh_{ID_{i\longrightarrow j}})^{d_{ID_{i\longrightarrow j}}^{-1}})$ with share verification information $\Omega_{ID_{i\longrightarrow j}}$, share key $Sh_{ID_{i\longrightarrow j}}$, and the user's private key d_{ID_i}. The effective of this verification can be proved in Claim 2.

(b) The user then recovers $K_{ID_{i\longrightarrow j}} = Sh_{ID_{i\longrightarrow j}}^{d_{ID_i}^{-1}} = h_{ID_i}^{a_k \cdot \lambda_{k_j} \cdot d_{ID_i}^{-1}} = g^{a_k \cdot \lambda_{k_j}}$ from $SK_{i\longrightarrow j}$ by his private key d_{ID_i}.

(c) Finally the user computes the decryption key $DK_{i\longrightarrow j} = \{K_{ID_{i\longrightarrow j}} = g^{a_k \cdot \lambda_{k_j}}, K_k' = g^{\alpha_k} \cdot g^{a_k \cdot t_k}, K_k'' = g^{t_k}, \Omega_{ID_{i\longrightarrow j}}, v_{R_{ID_{i\longrightarrow j}}}\}$.

4.4. Decryption of Data File. A_k is an authority set of the group U_k and $\{A_k \mid A_k \subseteq U_k, k \in \{1, \cdots, N\}\}$ are the union of the N authority sets for N groups.

(1) These authorized users of the N sets A_k compute the tag $Tag_{owner} = H(AttValue_{owner})$ of data file that they want to decrypt and then send it to cloud server.

(2) The cloud server receives the tag Tag_{owner}.

The cloud server verifies

$$BF[h_1(Tag_{owner})] = BF[h_2(Tag_{owner})] = \cdots = BF[h_o(Tag_{owner})] = 1, \tag{3}$$

using Tag_{owner} provided by these users. If it is satisfied, two decryption approaches can be adopted to get the plaintext: (a) decrypting by these authorized users and (b) outsourcing the decryption to the cloud. We describe them as follows:

(a) Decrypting by these authorized users.

After the authorized users A_k receive the corresponding data file $\langle BF_{datafile}, (M_k, \rho_k), C_0, C_k\rangle_{k=1\sim N}$ sent by the cloud server, one has the following:

(1) All the authority users from those authority sets A_k use their decryption keys $(Sh_{ID_{i\longrightarrow 1}}, v_{R_{ID_{i\longrightarrow 1}}}), \cdots, (Sh_{ID_{i\longrightarrow l_k}}, v_{R_{ID_{i\longrightarrow l_k}}})$ to verify the correctness of these decryption keys.

Firstly, it is recovered $\widehat{R}_k(x)$ from $(v_{R_{ID_{i\longrightarrow 1}}}, \cdots, v_{R_{ID_{i\longrightarrow l_k}}})$ using Berlekamp algorithm. Then, the users of the group U_k verify $\widehat{R}_k(\mu(Sh_{ID_{i\longrightarrow j}}, ID_{i\longrightarrow j})) = v_{R_{i\longrightarrow j}}$ for every decryption key $DK_{i\longrightarrow j}$. If it is unequal then $Sh_{ID_{i\longrightarrow j}}$ is a fake share key. The user $ID_{i\longrightarrow j}$ is added to the list of cheaters L_k. Moreover, every group can identify the cheaters of the group in this way.

(2) If there are no cheaters in all of the authority users, then these authority users recover the blind factor of data $\mathscr{M}$ as follows.

Firstly, for every authority set, there exists constant $\beta_k \in Z_p^n$ satisfying $\sum_{j=1}^{l_k} \lambda_{k_j} \cdot \beta_k = s_k$, where $k = 1 \cdots N$. Then, the blind factor of data $\mathcal{M}$ is computed as

$$\begin{aligned}
&\frac{\prod_{k=1}^{N} e\left(C_k, K_k'\right)}{\prod_{k=1}^{N}\left(\prod_{\substack{j=1 \\ ID_{i\to j}\in Ak}}^{l_k}\left(e\left(Sh_{ID_{i\to j}}, K_k''\right)^{\beta_k}\right)\right)} \\
&= \frac{\prod_{k=1}^{N} e\left(g^{s_k}, g^{\alpha_k} g^{a_k \cdot t_k}\right)}{\prod_{k=1}^{N}\left(\prod_{\substack{j=1 \\ ID_{i\to j}\in A_k}}^{l_k} e\left(g^{a_k \cdot \lambda_{k_j}}, g^{t_k}\right)^{\beta_k}\right)} \\
&= \prod_{k=1}^{N} \frac{e\left(g^{s_k}, g^{\alpha_k}\right) \cdot e\left(g^{s_k}, g^{a_k \cdot t_k}\right)}{\left(e\left(g, g\right)^{\sum_{\substack{j=1 \\ ID_{i\to j}\in A_k}}^{l_k} a_k \cdot \lambda_{k_j} \cdot \beta_k \cdot t_k}\right)} \\
&= e\left(g, g\right)^{\sum_{k=1}^{N} \alpha_k \cdot s_k}
\end{aligned} \tag{4}$$

Finally, $\mathcal{M} = C_0 / e(g, g)^{\sum_{k=1}^{N} \alpha_k \cdot s_k}$.

(b) Outsourcing the decryption to the cloud.

In this situation, these authorized users first generate the transformation key $TKCloud_k$ in groups for the cloud before they are outsourcing the decryption and then obtaining group decryption GK_k, where $k \in \{1, \cdots, N\}$. To generate $TKCloud_k$ and GK_k, the authorized user group A_k chooses a random value $z_k \in Z_p$ and computes the transformation key $TKCloud_k$ as $TKCloud_k = (Sh_{ID_{i\to 1}})^{1/z_k}, \cdots, (Sh_{ID_{i\to l_k}})^{1/z_k}, (K')^{1/z_k^2}, (K'')^{1/z_k}$ and outputs the group decryption key $GK_k = z_k$, for $k \in \{1, \cdots, N\}$. We allow these authorized users themselves to generate the transformation key in group, which is more flexible. Then they send transformation key $\langle TKCloud_k \rangle_{k=1\cdots N}$ to the cloud server for outsourced decryption.

The cloud computes the following equation using $\langle TKCloud_k \rangle_{k=1\sim N}$:

$$\begin{aligned}
\widetilde{C}_0 &= \frac{e\left(C_k, \left(K_k'\right)^{1/z_k^2}\right)}{\prod_{\substack{j=1 \\ ID_{i\to j}\in Ak}}^{l_k}\left(e\left(\left(Sh_{ID_{i\to j}}\right)^{1/z_k}, \left(K_k''\right)^{1/z_k}\right)^{\beta_k}\right)} \\
&= \frac{e\left(g^{s_k}, \left(g^{\alpha_k} g^{a_k \cdot t_k}\right)^{1/z_k^2}\right)}{\prod_{\substack{j=1 \\ ID_{i\to j}\in A_k}}^{l_k} e\left(g^{a_k \cdot \lambda_{k_j}}, g^{t_k}\right)^{\beta_k \cdot (1/z_k^2)}} \\
&= \frac{e\left(g^{s_k}, g^{\alpha_k}\right)^{1/z_k^2} \cdot e\left(g^{s_k}, g^{a_k \cdot t_k}\right)^{1/z_k^2}}{\left(e\left(g, g\right)^{\sum_{\substack{j=1 \\ ID_{i\to j}\in A_k}}^{l_k} a_k \cdot \lambda_{k_j} \cdot \beta_k \cdot t_k \cdot (1/z_k^2)}\right)} \\
&= e\left(g, g\right)^{\alpha_k \cdot s_k \cdot (1/z_k^2)}
\end{aligned} \tag{5}$$

and then it sends $\widetilde{C}_0$ to the user groups A_k, for $k \in \{1, \cdots, N\}$. After receiving $\widetilde{C}_0$, all the user groups A_k compute message $\mathcal{M}$ as $C_0 / \prod_{k=1}^{n} (\widetilde{C}_0)^{z_k^2} = C_0 / \prod_{k=1}^{n} (e(g, g)^{\alpha_k \cdot s_k \cdot (1/z_k^2)})^{z_k^2} = \mathcal{M}$.

5. Security Analysis

5.1. Provable Security. In the proposed scheme, we utilize a symmetric encryption to hide the message data and use secret sharing based on general access structure to share the session key with users in every group. Since the security of secret sharing based on general access structure does not rely on any computational complexity assumption, it is unconditionally secure. Therefore, the modification does not disclose any information, and the proposed scheme is chosen plaintext secure.

5.2. Privacy of the Data Owner's Personal Information. In order to protect privacy of the data owner, the data owner first computes the tag of the data file $Tag_{owner} = H(AttValue_{owner})$ with public hash function H and constructs the bloom filter $BF_{datafile} = BF(Tag_{owner})$ of the tag Tag_{owner}. Then the bloom filter $BF_{datafile}$ with the ciphertext is constructed as data file and uploaded to cloud server. To decrypt the ciphertext they want to access, the cloud users firstly compute the Tag_{owner} of the data file with public hash function H. Then they present the Tag_{owner} to the cloud server; the cloud finds the item of the data file by verifying the bloom filter $BF_{datafile}$. If it passes the verification, the ciphertext of the data file $\langle BF_{datafile}, (M_k, \rho_k), C_0, C_k \rangle_{k=1\sim N}$ is sent to users. Then the users decrypt the ciphertext to recover the plaintext with their decryption key. It is easy to see that the personal privacy of the owner can be protected from the cloud.

Claim 1. The correctness of the search result can be ensured if the rate of false search result is acceptable.

A false positive probability *PosiPro* exists when determining whether an element x belongs to a set because of the possible collisions in the hash functions. We can compute *PosiPro* as follows [15]:

$$PosiPro = \left(1 - \left(1 - \frac{1}{m}\right)^{kn} \cdot k\right) \approx \left(1 - e^{-kn/m}\right)^k, \tag{6}$$

where n is the number of elements in set S and m is the size of the bit array. Obviously, when $k = (\log 2)(m/n)$, the false positive probability *PosiPro* can be minimized to a negligible value, i.e., $(1/2)^k$.

5.3. Verifiability of the Share Key Distribution. In the following, we prove that the user can verify the correctness of share key distributed by the data owner.

Claim 2. Suppose a user is ID_i; if he accepts the share verification $\Omega_{ID_{i\to j}}$ from the owner, then there exists a unique value $Sh_{ID_{i\to j}}$ such that $e(\prod_{j=1}^{l_k} (\sigma_{ID_{i\to j}})_j^{m_{k_{ij}}}, g) = e(\widehat{g}, (Sh_{ID_{i\to j}})^{d_{ID_{i\to j}}^{-1}})$.

Table 2: Comparison of security properties.

Scheme	Collusion-resistant	Anonymous storage	Having verification property	Identifiable cheater
[2]	YES	NO	YES	NO
[4]	YES	NO	NO	NO
Ours	YES	YES	YES	YES

Suppose the owner distributes a share verification $\Omega'_{ID_{i\to j}} = \{\widehat{g}^{a_k \cdot z_1}, \cdots, \widehat{g}^{a_k \cdot z_{l_k}}\}$ to the user ID_i; if the user accepts the value $\Omega'_{ID_{i\to j}}$, then $e(\prod_{j=1}^{l_k} \widehat{g}^{a_k \cdot z_1} \cdots \widehat{g}^{a_k \cdot z_{l_k}}, g) = e(\widehat{g}, (Sh_{ID_{i\to j}})^{d^{-1}_{ID_i \to j}}), e(\prod_{j=1}^{l_k} \widehat{g}^{a_k \cdot z_1} \cdots \widehat{g}^{a_k \cdot z_{l_k}}, g) = e((\widehat{g})^{a_k \cdot \sum_{j=1}^{l_k} z_j \cdot m_{k_{ij}}}, g) = e(\widehat{g}, g^{\sum_{j=1}^{l_k} a_k \cdot z_j \cdot m_{k_{ij}}}) = e(\widehat{g}, (Sh_{ID_{i\to j}})^{d^{-1}_{ID_i \to j}})$, which leads to $Sh_{ID_{i\to j}} = h^{\sum_{j=1}^{l_k} a_k \cdot z_j \cdot m_{k_{ij}}}_{ID_{i\to j}} = h^{a_k \cdot \lambda_{k_j}}_{ID_{i\to j}}$; that is to say $\lambda_{k_j} = z_j \cdot m_{k_{ij}}$.

Therefore, the share key can be verified by the user.

5.4. Identification of the Cheater. The RS code is employed here to prevent the cheaters from changing the polynomial $R(x)$, which is used to test the validity of the shares, due to the fact that RS code has the ability to perform error correction. In other words, a polynomial $R(x)$ of t degree is uniquely determined by $R_{i1}, \cdots, R_{ik}$ if and only if $k \geq 3t + 1$, even if there are some fake shares R_{ij} they cannot prevent the correct reconstruction of $R(x)$. Thus, we have the following conclusion.

Claim 3. If $t \leq (k - 1)/3$ in every group $U_{\varphi(i)}$, then the proposed scheme is a (t, ε) cheater identifiable data sharing scheme that no cheater can succeed in cheating without being identified with probability better than $1/q$, where t is the number of the cheaters and k is the threshold for every group.

5.5. Mutual Supervision between Groups

Claim 4. The proposed scheme is collision-resistant and mutual supervision between different groups.

The proposed scheme not only identifies the cheater in group but also can achieve mutual supervision between groups, because the secret is distributed to each group, if and only if all of the N groups present the correct secret share, then the plaintext is recovered correctly. If some users present a fake share in any authorized user set, it would cause the group share error, which leads the decryption process to failure.

6. Performance Analyses

In this section, we briefly compare our scheme with some other classical data sharing schemes, like Yu scheme [2] and Dong scheme [4]. Yu scheme is relaying on KP-ABE, while Dong scheme is based on CP-ABE. In our scheme, t_k can also be seen as the attribute that involved in the group U_k. These schemes have applications in healthcare or library scenarios to share data. Besides, our scheme is more suitable for medical supervision scenario, especially when evidence is needed like EMR in medical disputes or accident. Data confidentiality is achieved in all these schemes since the data owner stores the ciphertext of data file into cloud server, and the cloud servers are not able to learn the plaintext of any data file. The data decryption keys are not known by the cloud server in any one of these schemes [2, 4] as well as in our scheme, although the proxy reencryption key is given to the cloud server in [2]. The comparison of our scheme with the schemes in [2, 4], regarding the security properties, is summarized in Table 2.

In order to make the comparison fair and meaningful, when comparing the computational cost in the decryption phase, we consider the general situation that all users have participated. In case when the decryption is partially outsourced to the cloud, this will further reduce the computational cost for the users obviously.

6.1. Dynamic Operation. The dynamic operations such as user addition/revocation and file creation/deletion are processed in a similar way as in all these schemes, then the operation are introduced briefly.

(*1) File Operation*. There are three operations for file operation: file creation, file deletion, and file updating. In these operations, the data owner has the right to delete the new file in [2, 4] and ours. He makes a unique tag and defines the access policy or attributes set for file creation. For file updating, the data owner ought to rearrange the access policy in our scheme and in [4], while updating the file's attribute i in ciphertext and the proxy reencryption key in scheme [2].

(2) User Operation. There are two operations for user operations: new user grant and user revocation. Similarly, the data owner assigns the corresponding secret key to the new user according the type of the scheme, KP-ABE or CP-ABE. The data owner may revoke the access privileges from some users. It has been a great challenge to achieve an effective user revocation.

In most existing schemes, it can take a direct manner in which the data owner reupdates the access policy, reencrypts the relevant files, and distributes the renewed keys to the nonrevoked users via the cloud server. This method is applicable in [2, 4] and our scheme. Based on the above reasons, the data owner needs to guarantee that all the operations are processed faithfully by the cloud servers.

6.2. Computation Complexity. In this section, we analyze and compare the computation overhead of the proposed scheme with [2, 4], considering the encryption and decryption

Table 3: Comparison of computation complexity.

Scheme	Encryption (Data owner)	Decryption (User)	Key Generation
[2]	$O(\lvert I_u\rvert)$	$O(\max(\lvert I_M\rvert, \lvert I_u\rvert))$	$O(\lvert I_M\rvert)$
[4]	$O(\lvert I_M\rvert)$	$O(\lvert I_M\rvert)$	$O(\lvert I_u\rvert)$
Ours	$O(N)$	$O(\lvert U\rvert)$	$O(1)$

operation. In the proposed scheme, the main computational cost involved in encryption and decryption algorithms are pairing $e(g, g)$ and scalar multiplication. The ciphertext of the proposed scheme is $C = \{C_0 = e(g, g)^{\sum_k \alpha_k \cdot s_k}, C_1 = g^{s_1}, \cdots, C_N = g^{s_N}\}$. Pairing is the most expensive operation. For each different file, data owner only needs to calculate $e(g, g)$ once at the beginning. Thus, we do not consider the overhead of pairing operation in the computational complexity when comparing the proposed scheme with those in [2, 4]. In the computational complexity analysis, we only take into account scalar multiplication operation. During encrypting, all encryption operations are at the data owner's side. The data owner needs to do N scalar multiplication for C_0 and N scalar multiplications for C_k; thus, the owner should take $2N$ scalar multiplication in total for encryption. Then the computation complexity of encryption is $O(N)$.

In the decryption stage, to recover ciphertext, the user needs at most $(2|N| + |U|)$ scalar multiplications to calculate

$$\prod_{k=1}^{N} \frac{e(g^{s_k}, g^{\alpha_k}) \cdot e(g^{s_k}, g^{a_k \cdot t_k})}{e(g, g)^{\sum_{j=1}^{l_k} a_k \cdot \lambda_{k_j} \cdot \beta_k \cdot t_k}_{ID_{i \to j} \in A_k}}. \tag{7}$$

Thus, the computation complexity of decryption is at most about $O(|U|)$.

In the key generation stage, it needs one scalar multiplication to calculate $Sh_{ID_{i \to j}} = h_{ID_i}^{a_k \cdot \lambda_{k_j}}$ for each user ID_i and two scalar multiplications to calculate $K_k' = g^{\alpha_k} \cdot g^{a_k \cdot t_k}$ and $K_k'' = g^{t_k}$ for each group U_k. Therefore, the computational complexity of key generation is at most $O(1)$.

For the cloud server, the main computational overhead is caused by the execution of tag testing algorithm by Bloom filter hash function. So the computation complexity for cloud server is $O(1)$.

In [4], the data owner needs to do two scalar multiplications to calculate $C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x}$, one scalar multiplication for $C_{2,x} = g_1^{r_x}$, and two for $C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x}$. Therefore, the data owner needs at most $5|I_M|$ scalar multiplications. Thus, the computation complexity of encryption is $O(|I_M|)$. To recover the ciphertext, the user needs another $2|I_M|$ scalar multiplications at most to calculate $\prod_x \{e(g_1, g_1)^{\lambda_x} e(H(ID), g_1)^{\omega_x}\}$, so the time complexity is also at most $O(|I_M|)$. The time complexity for key generation is $O(|I_u|)$.

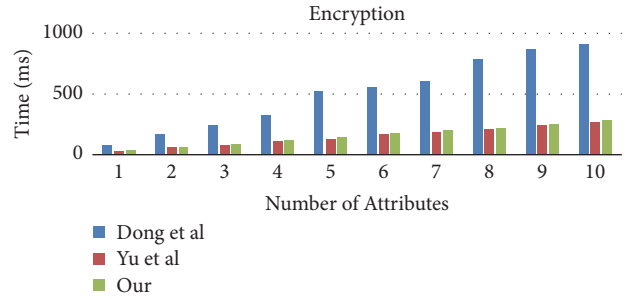

Figure 3: Comparison of encryption performance with 10 attributes.

While in [2], the data owner needs to do one scalar multiplications to calculate $\widetilde{E} = M \cdot e(g, g)^{ys}$ and one scalar multiplication for $E_i = g^{t_i \cdot s}$. Therefore the computation complexity of the data owner for encryption is $O(|I_u|)$. To recover the ciphertext one has to compute $e(E_i, sk_i) = e(g, g)^{p_i(0)s}$ for each leaf node firstly. Then, it aggregates these pairing results in the bottom-up manner using the polynomial interpolation technique. Finally, it recovers the blind factor $Y^s = e(g, g)^{ys}$ and outputs the message M if only if attributes I satisfy access tree T. So the time complexity for decryption is about $O(\max(|I_M|, |I_u|))$. And the time complexity for generation of the key $SK = \{sk_i \mid sk_i = g^{p_i(0)/t_i}\}_{i \in I_M}$ is $O(|I_M|)$.

The computational complexity of our scheme, as well as [2, 4], is given in Table 3, where I_u denotes the attributes of the user, I_M denotes the attributes of access structure, set U represents the universal users, and N is the number of the partitioned groups in our scheme.

6.3. Experiment Results. The evaluation is conducted through experiment evaluating the time cost of the proposed scheme on a computer with Windows7 Intel i5-4590S -3.00GHz CPU, and 4-GB RAM. All results presented here are the average value in 100 different trials.

6.3.1. The Overhead of Encryption Algorithm. In our scheme, the encryption is to calculate $C_1, \cdots, C_N$. In [2], the calculation of ciphertext C is based on $\widetilde{E}$ and $\{E_i\}_{i \in I}$. And the calculation of ciphertext C in scheme [4] is based on $C_{1,x}, C_{2,x}, C_{3,x}$. Let the number of attributes $|I_u|$ equals the number of users N, and then the encryption speed of our scheme and other schemes with the number of the attribute is to 10 and to 50 is given, respectively, in Figures 3 and 4.

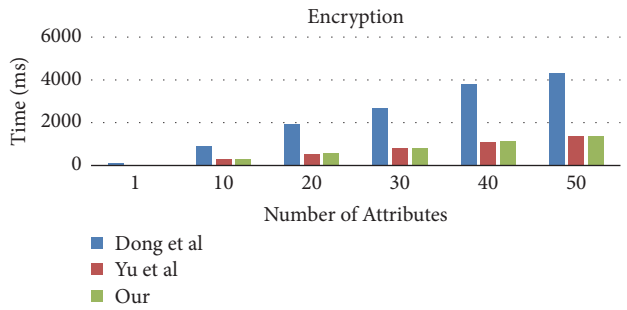

Figure 4: Comparison of encryption performance with 50 attributes.

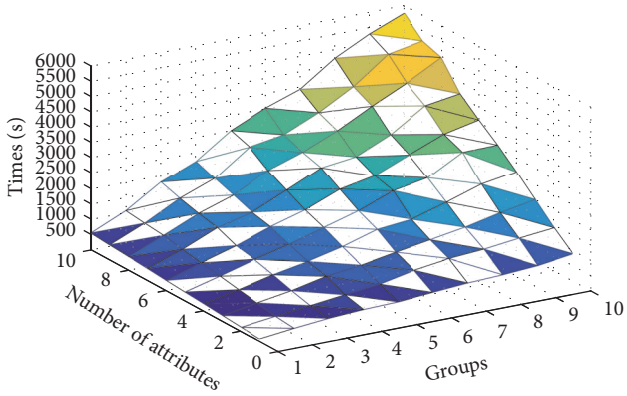

Figure 5: The decryption performance of our scheme.

From Figures 3 and 4, we can see that the encryption cost increases linearly with the attributes in the three schemes, and our scheme has almost the same cost as [2] and it is much lower than [4].

6.3.2. The Overhead of Decryption Algorithm. In our scheme, the decryption is to compute $\prod_{k=1}^{N} e(C_k, K'_k) / \prod_{k=1}^{N} \left(\prod_{\substack{j=1 \\ ID_{i\to j} \in Ak}}^{l_k} (e(Sh_{ID_{i\to j}}, K''_k)^{\beta_k}) \right)$. As we see the cost of decryption depends on the number of groups and the user number in each group. The decryption overload about 10 groups and 10 users in every group of our scheme is showed in Figure 5.

The decryption overload of [2] is to compute the pairing operation which is not only due to the number of the attributes but also due to intermediate node, the size of the concrete tree structure. If the structure of the tree is a large number, then the overload will be very large. Here, we only give the comparison of decryption overload between our scheme and [4], where the number of users in our scheme is the same as the number of attributes in [4]. Every star in Figure 6 is denoted as a number of the groups in brackets under the same number of the attributes in our scheme. From Figure 6, we can see that the more groups, the greater the consumption.

When all users are in one group in our scheme, the overhead of decryption are showed in Figures 7 and 8, where

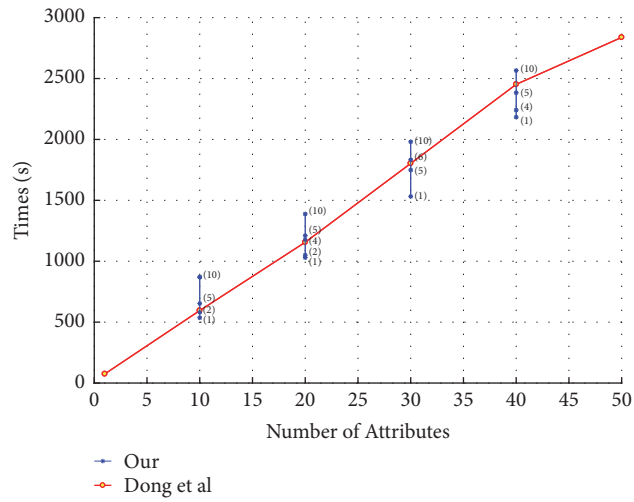

Figure 6: Decryption performance of the proposed scheme and [4].

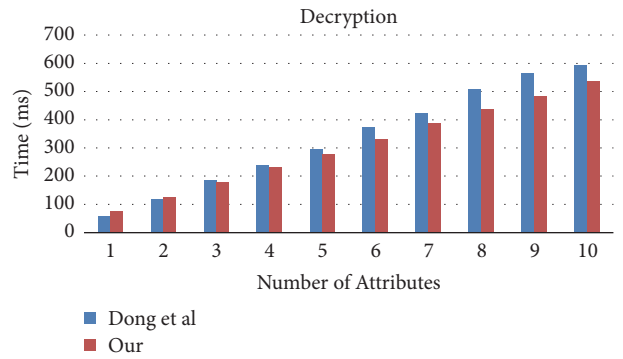

Figure 7: Comparison of decryption performance with 10 attributes.

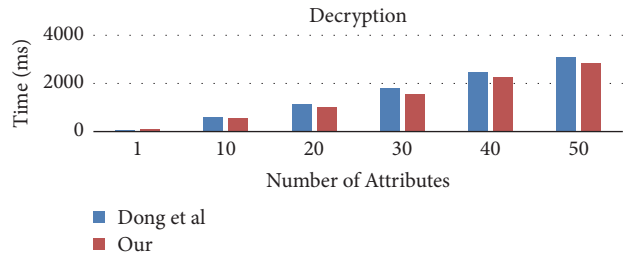

Figure 8: Comparison of decryption performance with 50 attributes.

attributes of access structure are up to 10 and 50, respectively. Then it can be seen that our scheme overhead is less than [4].

6.3.3. The Overhead of Key Generation Algorithm. The key generation algorithm is to compute the power exponent in all of three schemes. In order to simplify the comparison, we take all users in one group, then key generation overloads of three schemes are showed in Figure 9. From the Figure 9, it can be seen that the overload in our scheme is much less than [4] and a little more than [2].

Table 4: Comparison of communication costs.

Scheme	Communication costs
[2]	$\lvert I\rvert + 2\log\lvert I\rvert + (\lvert I\rvert + 1)\log\lvert\mathbb{G}_1\rvert + \log\lvert\mathbb{G}_2\rvert + data$
[4]	$\lvert I\rvert^2 + \log\lvert I\rvert + (2\lvert I\rvert + 1)\log\lvert\mathbb{G}_1\rvert + (\lvert I\rvert + 1)\log\lvert\mathbb{G}_2\rvert + data$
Ours	$\log\lvert\mathbb{G}_2\rvert + N\cdot\log\lvert\mathbb{G}_1\rvert + 3N\cdot\log\lvert\mathbb{G}_2\rvert + N^2\cdot\log\lvert\mathbb{G}_2\rvert + \log\lvert q\rvert + data$

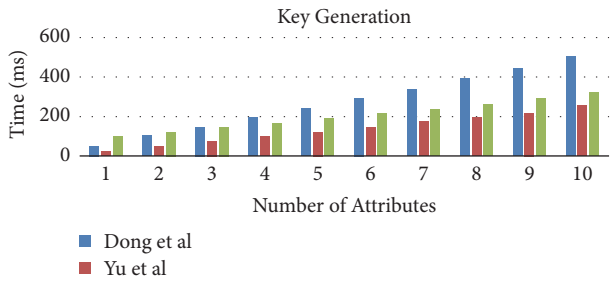

Figure 9: Comparison of key generation performance with 10 attributes.

6.4. Communication Cost. In our scheme, the communication cost is mainly attributed to the encrypted data and key distribution transmission. The encrypted data is sent by the data owner to the cloud: the value of $C_0 = e(g,g)^{\sum_k \alpha_k \cdot s_k}$ and $C_1 = g^{s_1}, \cdots C_N = g^{s_N}$ requires $(\log\lvert\mathbb{G}_2\rvert + N\cdot\log\lvert\mathbb{G}_1\rvert)$ bits. The share keys are sent by the data owner to the users: the value $Sh_{ID_{i\to j}}, K_k', K_k''$ for every i requires $3N\cdot\log\lvert\mathbb{G}_2\rvert, \Omega_{ID_{i\to j}}$ at most requires $N^2\cdot\log\lvert\mathbb{G}_2\rvert$ and $v_{RID_{i\to j}}$ requires $\log\lvert q\rvert$ bits. Thus the communication cost of the share key from owner to users is given by $3N\cdot\log\lvert\mathbb{G}_2\rvert + N^2\cdot\log\lvert\mathbb{G}_2\rvert + \log\lvert q\rvert$. The private key is usually a few hundred bits, and in general, it does not need to be compressed. We need to assume that before the cloud environment is established, the private key is initialized in advance, and each participant can securely store and use the private key. Thus the whole communication cost of the protocol is given by $\log\lvert\mathbb{G}_2\rvert + N\cdot\log\lvert\mathbb{G}_1\rvert + 3N\cdot\log\lvert\mathbb{G}_2\rvert + N^2\cdot\log\lvert\mathbb{G}_2\rvert + \log\lvert q\rvert$ + data. The communication expenses comparison between our scheme, KP-ABE-based schemes, and CP-ABE-based schemes is shown as Table 4. We can see that the communication cost of our scheme is nearly the same as CP-ABE-based schemes and our scheme and [4] is slightly more than KP-ABE-based schemes. However, in practice, the file is described by just a limited attributes or shared with limited users. In addition, even though the order of cyclic group G is large, $\log\lvert G\rvert$ bits is far less than the file size (data). In other words, the extra communication cost can be ignored.

7. Application to Secure Medical Information Sharing Scene

In personal health medical information environment, like personal medical information, medical record information of a person is cumulated consistently during his life; he will have a lot of contact with nurses and doctors over his life. From perspective of the patient, he is the data owner. When his health medical record is stored in the cloud server, he also wishes to control his medical data and he needs to specify who can to access his information; those users are called authorized users. As shown in Figure 10, these authorized users might be some friends, specialists, nurses, and public security investigators. To ensure impartiality and fairness, to prevent tampering, forgery, and other illegal acts, the access of medical record data of the owner should be carried out under the above different groups' supervision. And scheme should have the properties of data confidentiality and privacy protection and cheater identification. To achieve this goal, a privacy protection approach is taken to use bloom filter, to hide some personal information that is not closely related to health conditions of the patient, such as name, gender, telephone number, ID card number, family address, and property, when medical record of owner is stored on cloud. Moreover, since each group has a group secret, data's access is carried out under an effective supervision mechanism according to the portioned groups. Besides, it can be made sure that the participants conspiring or deceive can be found and identified applying an error correction function of RS encoding technique. In summary, the proposed scheme is helpful for patient to achieve flexible and supervised control on his case file stored on cloud server.

8. Conclusion

In this paper, a personal medical information privacy protection scheme in the cloud was proposed, which can be used to set the electronic medical records system up for patients efficiently. The proposed scheme has flexible data access control through combing the techniques of the secret sharing methods and symmetric encryption. The performance analysis shows that the proposed scheme has low overhead and high efficiency. In this proposed scheme, we use RS encoding method to identify the dishonest user. It means there are not too much misbehave users in every group. As indicate in Section 2.5, this scheme has the capability of identifying up to t cheaters under the condition $(k-1)/3 \geq t$. Hence, in the future works, we will investigate how to remove this condition and to achieve more efficiency of recognition of the dishonest user. Moreover, we will investigate how to achieve efficient data file updated flexibly and how to process multifile convergence in batches.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

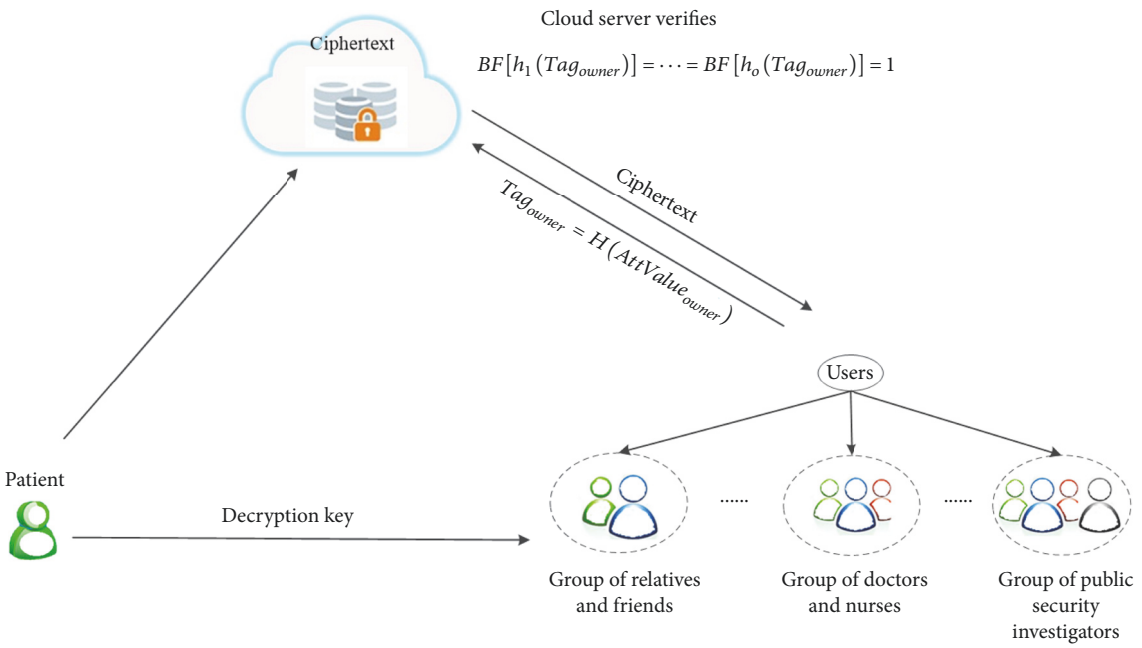

Figure 10: The system model for personal medical information privacy protection system.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Key R&D Program of China (no. 2017YFB0802000), the National Natural Science Foundation of China (61572303, 61772326, 61802241, 61802242, and 61872289), National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20180217), the Foundation of State Key Laboratory of Information Security (2017-MS-03), the Provincial Natural Science Foundation Research Project of Shaanxi (no. 2017JQ6029), the Shaanxi Provincial Department of Education Special Scientific Research Project (no. 16JK1109), and the Doctoral Scientific Fund Project of Shaanxi University of Science and Technology (BJ11-12).

References

[1] M. Kallahalla, E. Riedel, R. Swaminathan et al., "Scalable secure file sharing on untrusted storage," in *Proceedings of the FAST'03 Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pp. 29–42, 2003.

[2] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, March 2010.

[3] 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 '06)*, pp. 89–98, November 2006.

[4] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Computers & Security*, vol. 42, pp. 151–164, 2014.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, May 2007.

[6] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Public Key Cryptography (PKC '11)*, pp. 53–70, Springer, Berlin, Germany, 2011.

[7] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Advances in Cryptology—CRYPTO 2006*, vol. 4117 of *Lecture Notes in Computer Science*, pp. 209–307, Springer, Berlin, Germany, 2006.

[8] Beimel., *Secure schemes for secret sharing and key distribution [Ph.D. thesis]*, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[9] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electronics & Communications in Japan*, vol. 72, no. 9, pp. 56–64, 1989.

[10] J. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," *On Advances in Cryptology*, vol. 403, pp. 27–36, 1988.

[11] M. Karchmer and A. Wigderson, "On span programs," *The Eighth Annual Structure in Complexity Theory*, pp. 102–111, 1993.

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

[13] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the International Conference on Applied Cryptography & Network Security*, vol. 5037, pp. 111–129, 2008.

[14] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2012*, pp. 18-19, Republic of Korea, May 2012.

[15] S. Jiang, X. Zhu, and L. Wang, "EPPS: Efficient and privacy-preserving personal health information sharing in mobile healthcare social networks," *Sensors*, vol. 15, no. 9, pp. 22419–22438, 2015.

[16] R. J. McEliece and D. V. Sarwate, "On sharing secrets and Reed-Solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583-584, 1981.

[17] S. Obana, "Almost optimum t-cheater identifiable secret sharing schemes," in *EUROCRYPT*, vol. 6632, pp. 284–302, 2011.

[18] H. Hoshino and S. Obana, "Cheating detectable secret sharing scheme suitable for implementation," in *Proceedings of the 4th International Symposium on Computing and Networking, CANDAR 2016*, pp. 623–628, Japan, November 2016.

[19] Z. Chen, S. Li, Q. Huang, J. Yan, and Y. Ding, "A joint random secret sharing scheme with public verifiability," *International Journal of Network Security*, vol. 18, no. 5, pp. 917–925, 2016.