Shen *et al. EURASIP Journal on Wireless Communications and Networking* (2019) 2019:277
https://doi.org/10.1186/s13638-019-1614-y

EURASIP Journal on Wireless Communications and Networking

RESEARCH **Open Access**

# Multi-security-level cloud storage system based on improved proxy re-encryption

Jinan Shen[1,2*], Xuejian Deng[1] and Zhenwu Xu[1]

**Abstract**

Based on the characteristics and data security requirements of the cloud environment, we present a scheme for a multi-security-level cloud storage system that is combined with AES symmetric encryption and an improved identity-based proxy re-encryption (PRE) algorithm. Our optimization includes support for fine-grained control and performance optimization. Through a combination of attribute-based encryption methods, we add a fine-grained control factor to our algorithm in which each authorization operation is only valid for a single factor. By reducing the number of bilinear mappings, which are the most time-consuming processes, we achieve our aim of optimizing performance. Last but not least, we implement secure data sharing among heterogeneous cloud systems. As shown in experiment, our proposed multi-security-level cloud storage system implements services such as the direct storage of data, transparent AES encryption, PRE protection that supports fine-grained and ciphertext heterogeneous transformation, and other functions such as authentication and data management. In terms of performance, we achieve time-cost reductions of 29.8% for the entire process, 48.3% for delegation and 47.2% for decryption.

**Keywords:** Privacy preserving, Proxy re-encryption, Fine-grained, Cloud storage

## 1 Introduction

### 1.1 Motivation

Cloud computing has provided very important support for information use by government agencies and enterprises. Security issues have restricted the development of public cloud services[1], and in particular, the problem of how to protect the privacy of users has become a significant factor in their reluctance to adopt cloud computing[2, 3]. Most existing solutions only focus on a single type of threat, and this has given rise to certain problems.

Recently, the protection of data privacy for users of cloud storage has become one of the most critical issues in public cloud applications and has impeded the rapid development of cloud services. There are still many weaknesses in the current methods for the protection of private user data in cloud environments [4, 5]. Theoretical research has developed many algorithms for cloud storage security requirements, such as distributed systems with information flow control (DIFC) [6], attribute-based encryption (ABE) [7], proxy re-encryption (PRE) [8], fully homomorphic encryption [9], ciphertext search algorithms, and so on. However, the most common way to share stored ciphertext is to download the ciphertext data, decrypt it into plaintext, and then re-encrypt it before sending it to users to share. This process consumes a lot of network and computing resources and loses the advantage of cloud storage [10, 11].

With regard to the security issues around the privacy of users' data in the cloud environment [6], and after investigation and study of the inadequacy of the existing solutions, we aim to develop a scheme that can protect data privacy in the cloud environment. This scheme should be able to resolve the core issues preventing users from trusting cloud services; in other words, we aim to allow the users themselves to have control over their data file. Simultaneously, from a practical point of view, basic fine-grained control must be implemented, and there is also a need to carry out optimization to achieve good performance and compatibility [12–14].

*Correspondence: jnshen@hust.edu.cn
[1]School of Information Engineering, Hubei Minzu University, Xueyuan Road, Enshi 445000, China
[2]School of Computing Science and Technology, Huazhong University of Science and Technology, Luoyu Road 1037, Wuhan 430074, China

### 1.2 Methodology

We observe that all of the existing methods of protecting a user's data in a cloud environment have limitations to a greater or lesser extent, and we therefore propose a scheme that can resolve all of these problems, including key control, fine-grained control, revoking of permissions, performance optimization, and compatibility.

Firstly, in our scheme, each user has a pair of identity-based encryption (IBE)-type private and public keys, and some users may also have another pair of public key encryption (PKE)-type private and public keys. These are all generated by a trusted third party, and all private keys are kept by users themselves. There is never a need to tell other users the private key. When a user updates data in the cloud, he or she can encrypt these data using the IBE-type public key and can compute a re-encryption key that does not leak the user's secret key. The proxy also cannot obtain the plaintext during re-encryption. We also add fine-grained control, since each time the user computes the re-encryption key, this applies only to the data of the designated type. The proxy cannot transform data of one type to a ciphertext using a re-encryption key of the wrong type.

Secondly, our scheme can transform IBE-type ciphertext to PKE-type ciphertext; this feature means that our scheme has good compatibility, and it is straightforward to integrate our scheme with existing systems. Using optimization, we can reduce the use of bilinear mapping, which is a major influence on the performance of an algorithm, and our scheme therefore gives acceptable performance in real-world systems. We will describe this optimization in a later section.

Thirdly, we explicitly describe how to change and revoke permissions; when users do not want others to use data that were previously shared, the permissions can be conveniently changed.

We integrate all of these capabilities into a single scheme. Our main design goal is to help users achieve fine-grained access control to the storage and sharing of files in a cloud environment. Specifically, we aim to ensure that data are under the control of data owners, rather than cloud service providers. We also aim to create a scheme that has excellent performance and has all the relevant functions, such as permission revoking, type changing, data searches, and so on, to be appropriate for a real-world system.

This paper proposes an improved PRE algorithm to support fine-grained control and ciphertext heterogeneous transformation. A multi-security-level cloud storage system is designed that involves AES symmetric encryption and other algorithms. The user's private data are protected, while control over the data remains in the hands of the user.

This paper is organized as follows. In Section 2, we introduce related work on data protection using PRE in the public cloud. We carry out a formal analysis of the modeling and the macro definition of our scheme in Section 3. In Section 4, we introduce the associated implementation issues, including the system architecture and feature model. In Section 5,we conducted a safety analysis of our scheme. Section 6 presents an evaluation of our scheme. Finally, Section 7 concludes the paper and discusses future work.

## 2 Related work

Encryption is one of the most commonly used methods for traditional privacy data protection. Typical research results for cloud data security protection methods based on cryptography algorithms can be divided into symmetric and public key encryption classes, according to the type of encryption used.

Based on a symmetric encryption system, Li et al. [7] realized secure storage of cloud data and isolation of encrypted cloud data by adopting a method involving multiple keys and file partitioning. Sookhak et al. [8] combined the “lazy undo”, “keychain” and “broadcast encryption” operations to achieve more secure updating and permission transformation of cloud data. Liu et al. proposed the TrustStore method and introduced a special key management service provider to realize different forms of key encryption for different files. These authors also verified the integrity of cloud data by creating hash checking information [9]. Although this scheme solved the problem of secure storage and access control over cloud data to a certain extent, symmetric encryption lacks flexibility when used in key management and an encryption algorithm. Therefore, in a cloud computing environment with fine-grained multi-tenant dynamic sharing and large-scale networks, this type of method has drawbacks.

In an asymmetric encryption system, the core idea of ABE is to realize public key encryption of data by using the attributes of the data as the public key. In addition, the Boolean paradigm of data attributes can be used to realize fine-grained access control based on cryptography. Wu proposed a mixed scheme using ABE and symmetric encryption, and achieved secure encrypted storage and fine-grained access control for cloud data [15]. The work in [16] uses an ABE scheme to realize the secure sharing of health documents in the cloud. In addition to ABE, PRE is also used to implement secure storage and access control for cloud data. The core idea of PRE is that a third party can be authorized to re-encrypt the encrypted stored data to achieve secure access to the encrypted data [17]. The fine-grained scheme proposed by Zhang is based on a pre-identity scheme with a pre-interactive conditional scheme and solves the problem of the early pre-scheme being unable to realize fine-grained access control over the encrypted cloud data [18]. Jiang and Guo [19] proposed a PRE scheme for realizing the confidentiality of the data access control conditions of the encryption cloud. Zuo [20] proposed a PRE scheme that can be used one way and re-encrypted many times, thus realizing multi-tenant sharing of cloud data that can be transmitted. Li et al. [21] proposed an attribute-based, fine-grained authorization keyword search scheme for cloud computing, and Ding et al. [22] proposed a multi-keyword fuzzy search scheme, thus realizing privacy protection for encrypted data in cloud.

A symmetric, searchable encryption algorithm was proposed by Shen et al. [23]. Xu et al. [24] improved this scheme and gave the security definition. This scenario applies to both the data searcher and the data producer, with the main drawback being that existing solutions find a compromise between performance and functionality. An asymmetric searchable encryption algorithm was proposed by Ni et al. [25] and was improved by [26]. The disadvantage of this scheme is that existing asymmetric searchable encryption algorithms require elliptic curve pairing, which performs worse than a hash function or block cipher. The fast asymmetric searchable encryption algorithm proposed by Masayuki [27] improves the performance of the asymmetric searchable encryption algorithm but is also vulnerable to a dictionary attack. Guo [28] proposed a multi-user, symmetric, searchable encryption algorithm that is mainly aimed at the scenario in which encrypted data are generated by a single producer and multiple parties search these data. This not only supports the encryption of indices and the generation of search tokens, but also allows the data owner to add and revoke search rights to the data for certain users.

No complete solution exists for the dynamic protection of the user's data privacy. Crypt DB uses an onion-based security scheme to ensure different levels of data privacy. At the same time, this method uses homomorphic encryption to enable ciphertext to directly perform some simple operations, such as query and comparison, ensuring that data with a high privacy level is stored in memory as ciphertext. Holomorphic encryption refers to any operation that can be performed on plaintext on encrypted data without decryption [29]. Gasti et al. [30] proposed a feasible method for full mathematically homomorphic encryption, and this represented a decisive breakthrough in this technology. At present, software algorithms cannot perform complex calculations using ciphertext, so in order to encrypt the data in memory, many researchers have used hardware to ensure the privacy of dynamic data. XOM [31] guarantees that private data are still stored in memory as ciphertext and that data are decrypted by special hardware when entering the CPU for calculation.

## 3 Multi-security-level cloud storage system

Due to the diversity of tenants in the cloud environment, different tenants have different requirements for the security protection level of data, and even the same tenant has different requirements for the different data they own. In the cloud, user data exist in two forms: dynamic and static data. Static data refers to the information that does not participate in calculation and is mainly used for mass storage and convenient access, such as documents, pictures, video, reports, materials and so on stored by users for a long time. Dynamic data refers to the data that needs dynamic verification or participation in calculation, such as database files, program files, configuration files, etc. Our scheme for the protection of users data privacy in cloud storage includes the following: (1) normal interface and security interface: normal interface is used when users access public data, and security interface is used when they access data with a certain level of security; (2) security-level-based encrypted storage for static data, by applying encryption policies with different strengths to different user data, thus ensuring the user data confidentiality and high performance for the whole system; (3) a fine-grained access control mechanism for static data that combines PRE algorithms to support fine granularity and ciphertext heterogeneous conversion, web services technology, and secure plugging in client to protect the user's privacy; (4) security-level-based segmentation storage for dynamic data that provides segmentation storage for different user data, based on sensitivity; and (5) secure search technology for dynamic data, thus providing secure search services without exposing the data information to the cloud platform. The scheme is illustrated in Fig. 1.

In this paper, we focus solely on static data. The functional architecture based on multiple security levels is shown in Fig. 2. The functions on the front-end nodes of the system includes: (1) a user authentication module, (2) a client security plug-in module, (3) a multi-security-level encryption module, and (4) a data storage module. The PRE server includes the PRE service module and the client security plug-in module and provides a client security interface for users.

### 3.1 Enhanced RBAC

In order to solve the data security problem of the medical cloud platform, it is necessary to combine the access control policy with the encryption mechanism to deal with the privacy leakage problem of data storage procedures in the cloud environment. On the one hand, the access control policy authenticates the user's identity, which can ensure the confidentiality of the data (but the data stored in plaintext can be accessed by CSP, which cannot protect the privacy of the data). On the other hand, the data can be encrypted by encryption mechanism, and access control

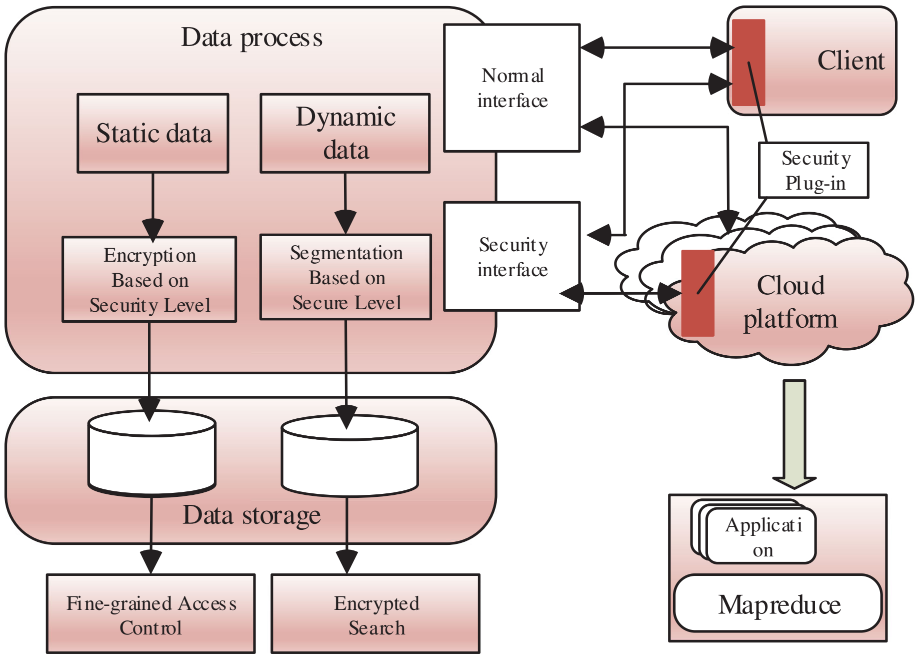

**Fig. 1** The multi-security-level cloud storage system

can be further strengthened by controlling the distributions of secret key. The access control model of medical cloud platform is shown in Fig. 3.

The platform realizes the access control and authorization for legal users of the system based on RBAC model. On the other hand, encryption is adopted to ensure the confidentiality of data in the cloud platform. The specific workflow is shown in Fig. 4.

The system management module is operated in the system initialization stage to realize the creation of users and roles and generate the corresponding public information $Mpp_R$ to the roles and the secret keys of the users and roles.

The user corresponding to each role in the Role Management Module Management System will establish the relationship between the user and the role to realize the addition and deletion of users in the role. The general permission distribution corresponding to the role is initialized by the system management.

$Mpp_R$, the public parameter that belongs to the date owners, is used to encrypt and store the date. Meanwhile, the date owners request the role management module to encrypt the data onto the corresponding role. The role management module determines which users are assigned to roles and which are excluded from its function to control user-to-role mapping.

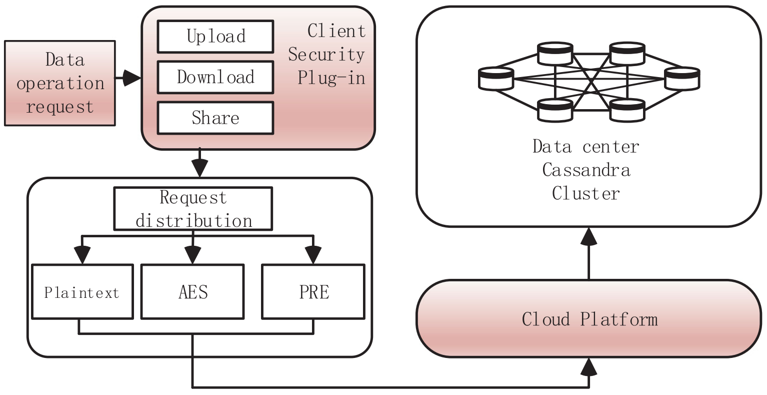

**Fig. 2** Architecture of the static data storage system

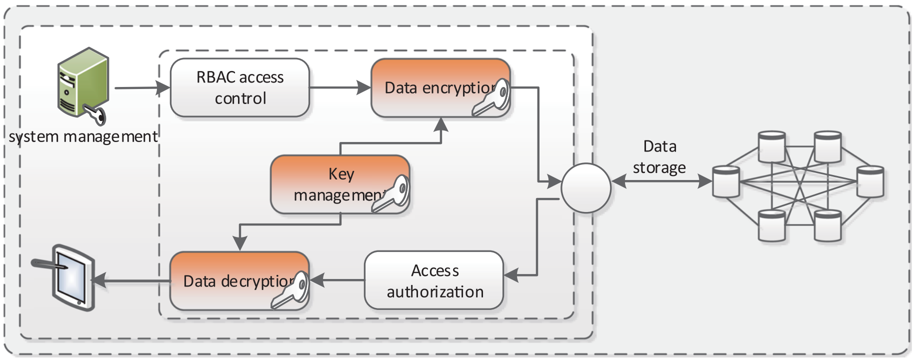

**Fig. 3** The access control enhancement model of medical cloud platform

The data requester uses the decryption key $DK_U$ provided by the system and the public information $Mpp_R$ of the role to decrypt the ciphertext from the cloud. It can be decrypted correctly when the data requester is a member of the data encryption or the role to which the data requester belongs inherit the ciphertext encryption role.

### 3.2 Design of the scheme

In this paper, the improved PRE algorithm supports fine-grained and ciphertext heterogeneous transformation. Based on the traditional identity-based PRE algorithm, fine-grained control and ciphertext heterogeneous transformation are added to allow fine-grained control of proxy authorization. Meanwhile, using the traditional

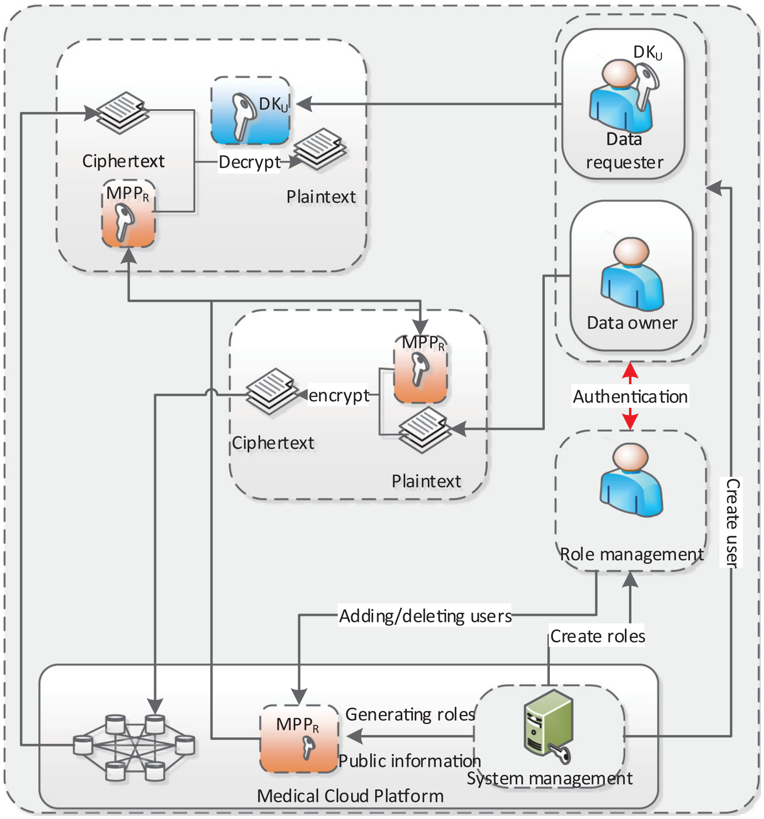

**Fig. 4** Access control model workflow in medical cloud platform

public key system, the re-encrypted ciphertext can be decrypted.

IBE algorithm consists of four main functions:

(1) $\text{Setup}_{\text{IBE}}(k)$: This algorithm generates system parameter pairs (params, mk), where $k$ is a security parameter, and params is a collection of system parameters. This parameter will be used in later functions. mk is the system master security key, and is related to the security of the overall system.
(2) $\text{KeyGen}_{\text{IBE}}(\text{mk}, \text{params}, \text{ID})$: mk is the system master security key, params is the system parameter set, and ID is the user's identity information. ID may include a mailbox, IP, ID number or other information. This algorithm generates the private key $\text{sk}_{\text{ID}}$ .
(3) $\text{Enc}_{\text{IBE}}(\text{ID},\text{params}, M, t)$: For plaintext data $M$ , we use the system parameter set params, identity information ID (public key), and a type value for fine-grained control $t$ to compute $M$, which corresponds to the original ciphertext.
(4) $\text{Dec}_{\text{IBE}}(sk_{\text{ID}}, \text{params}, C_{\text{ID}})$: If the proxy authorizer needs to decrypt and view encrypted data, the system parameter set params and the private key he holds can be used to decrypt the original cipher.

The Setup and KeyGen algorithms are executed by a trusted third-party key generation center (KGC) during the initialization phase of the system. The system parameters are generated and sent to the system through a secure channel for later use, while users can use their identity information to apply for the private key corresponding to the identity in the KGC.

The traditional public-key cryptographic algorithm consists of three functions ($\text{KeyGen}_{\text{PKE}}, \text{Enc}_{\text{PKE}}, \text{Dec}_{\text{PKE}}$):

(1) $\text{KeyGen}_{\text{PKE}}(k, \text{aux})$: $k$ is the security parameter, and aux is the corresponding parameter. This algorithm generates a pair of public and private key pairs (pk, sk), where pk is the public key, and sk is the private key.
(2) $\text{Enc}_{\text{PKE}}(\text{pk}, \text{aux},M)$: pk and aux are used to encrypt the plaintext $M$ to get the ciphertext $C_{\text{pk}}$ .
(3) $\text{Dec}_{\text{PKE}}(\text{sk}, \text{aux},C_{\text{PK}})$: Using sk and aux , the ciphertext $C_{\text{pk}}$ is decrypted to get the plaintext $M$.

This part of the function is used when converting the original ciphertext encrypted by the identity-based encryption algorithm into ciphertext that can be decrypted by the traditional algorithm, thus ensuring that the conversion process will not be affected by security problems.

The functions related to proxy re-encryption are $\text{KeyGen}_{\text{PRO}}()$ , ReEnc() , ReDec() and ChaType():

(1) $\text{KeyGen}_{\text{PRO}}(\text{sk}_{\text{ID}}, \text{sk}_i, \text{pk}_j, t, \text{params})$: This uses the system parameter set params, the fine-grained parameter $t$, the identity-based private key $\text{sk}_{\text{ID}}$ of the proxy authorizer, the private key $\text{sk}_i$ of the traditional public key system, and the public key $\text{pk}_j$ of the traditional public key system. This algorithm generates the PRE key $\text{rk}_{\text{ID}}$ and sends it to the PRE server for storage.
(2) $\text{ReEnc}(\text{rk}_{\text{ID}}, \text{params},C_{\text{ID}})$: This is the re-encryption function. Using $\text{rk}_{\text{ID}}$ and the corresponding system parameters, the original ciphertext $C_{\text{ID}}$ encrypted by IBE is re-encrypted into the ciphertext $C_{\text{pk}}$ of the traditional public key system.
(3) $\text{ReDec}(C_{\text{PK}}, \text{params}, \text{sk}_j)$: The decryption function of the re-encrypted ciphertext, using the private key $\text{sk}_j$ of the agent receiver and the corresponding system parameters, is used to decrypt the ciphertext $C_{\text{pk}}$ generated by re-encryption to the data plaintext $M$ .
(4) $\text{ChaType}(C_{\text{ID}}, \text{params}, t')$: This is the fine-grained dynamic change function. Using the changed fine-grained $t'$ and system parameters, the fine-grained information of ciphertext $C_{\text{ID}}$ is changed, and the fine-grained $t$ of ciphertext $C_{\text{ID}}$ is changed to the fine-grained $t'$ of ciphertext $C_{\text{ID}}^{'}$ .

## 3.3 Implementation of the scheme

The basic structure described in the previous section is used for a detailed overview of the proposed algorithm:

(1) $\text{Set}_{\text{IBE}}(k)$: This algorithm is run by the KGC, and generates a public parameter:

$$\text{params} = (G_1, G_T, p, g, H_1, H_2, \hat{e}, \text{pk}) \tag{1}$$

where $\text{pk} = g^a$,is the public key for the KGC. The security parameter $k$ is generated by a random algorithm in this system. $p$ is a prime, producing the multiplication groups $G_1$ and $G_T$ of order $q$. $g$ is a generator of $G_1$ , and a number $\alpha \in Z_q^*$ is randomly generated as the main safety parameter *mk*. A bilinear mapping function $\hat{e} : G_1 \times G_2 \to G_T$ and two hash function $H_1 : \{0, 1\}^* \to G_1, H_2 : \{0, 1\}^* \to Z_p^*$ are used.
(2) $\text{KeyGen}_{\text{IBE}}(\text{mk}, \text{params}, \text{id})$: When the user logs in, the KGC obtains the user's legitimate identity information id , and uses params and the main security parameter mk to generate the key pair $(\text{pk}_{\text{id}}, \text{sk}_{\text{id}})$ for the user, where

$$\text{pk}_{\text{id}} = H_1(\text{id}), \text{sk}_{\text{id}} = \text{pk}_{\text{id}}^{\alpha} \tag{2}$$

(3) $\text{KeyGen}_{\text{PKE}}(\text{params})$: This is run by the KGC when the user submits the application. Like the public and private keys based on identity type, $(\text{pk}'_{\text{id}}, \text{sk}'_{\text{id}})$ is generated by generating a random number $\gamma \in Z_p^*$ and calculating

$$\mathrm{pk}'_{\mathrm{id}} = g^{\mathrm{sk}'_{\mathrm{id}}}, \mathrm{sk}'_{\mathrm{id}} = \gamma \tag{3}$$

(4) $\mathrm{Enc}_{\mathrm{IBE}}(\mathrm{pk}_{\mathrm{id}}, \mathrm{sk}_{\mathrm{id}}, \mathrm{params}, m, t)$: This algorithm is executed when the data owner encrypts the data. The data owner uses his or her private key pair $(\mathrm{pk}_{\mathrm{id}}, \mathrm{sk}_{\mathrm{id}})$ based on identity type, type information $t$ for fine-grained control, and a random number $\gamma \in Z_p^*$ . The data are encrypted in plaintext to cipher $c = (c_1, c_2, c_3)$ , where

$$c_1 = g^r, c_2 = m \cdot \hat{e}(\mathrm{pk}_{\mathrm{id}}, \mathrm{pk})^{r \cdot H_2(\mathrm{sk}_{\mathrm{id}}) \| t}, c_3 = t \tag{4}$$

(5) $\mathrm{Dec}_{\mathrm{IBE}}(\mathrm{sk}_{\mathrm{id}}, \mathrm{params}, c)$: This function is used when the user decrypts a file that he or she has encrypted. Of course, the result could be obtained by requesting authorization and then re-encrypting the data, but the performance would be affected. When decrypting, the private key $\mathrm{sk}_{\mathrm{id}}$ of the data owner is entered, based on the identity type and the relevant system parameter params , and the cipher text params can then be decrypted to plaintext $m$ using the following calculation:

$$m = \frac{c_2}{\hat{e}(\mathrm{PK}_{\mathrm{id}}, \mathrm{pk})^{r \cdot H_2(\mathrm{sk}_{\mathrm{id}}) \| t}} \tag{5}$$

(6) $\mathrm{KeyGen}_{\mathrm{PRO}}(\mathrm{sk}_{\mathrm{id}}, \mathrm{sk}'_{\mathrm{id}}, \mathrm{pk}'_{\mathrm{id}}, t, \mathrm{params})$: When the data owner authorizes access, this step uses the $\mathrm{sk}_{\mathrm{id}_i}$ for the data owner based on the identity type, $\mathrm{sk}'_{\mathrm{id}_i}$ of the traditional type, $\mathrm{pk}'_{\mathrm{id}_j}$ of the agent receiver's traditional type and the fine-grained type $t$ , to calculate the agent from user $i$ to user $j$ to re-encrypt and re-encrypt the key.

$$\mathrm{rk}_{\mathrm{id}_i \to \mathrm{id}_j} = (t, \mathrm{sk}^{-H_2(\mathrm{sk}_{\mathrm{id}} \| t)} \cdot H_1(\mathrm{pk}_{\mathrm{id}_j}^{\prime \mathrm{sk}'_{\mathrm{id}_i}}), \mathrm{pk}'_{\mathrm{id}_i}) \tag{6}$$

(7) $\mathrm{ReEnc}(C_i, \mathrm{rk}_{\mathrm{id}_i \to \mathrm{id}_j}, \mathrm{params})$: This function is transparently executed on the server side, and encrypts the original ciphertext encrypted by user $i$ into ciphertext that user $j$ can decrypt. The ciphertext $c_i = (c_{i1}, c_{i2}, c_{i3})$ is encrypted by user $i$ and the agent of type $t$ from user $i$ to $j$ re-encrypt the key $\mathrm{rk}_{\mathrm{id}_i \to \mathrm{id}_j}$ and relevant system parameters are used as input, and the re-encrypted $c_j = (c_{j1}, c_{j2}, c_{j3})$ is the output after calculation.

$$\begin{cases} c_{j1} = c_{i1} \\ c_{j2} = c_{i2} \cdot \hat{e}\left(c_{i1}, \mathrm{sk}_{\mathrm{id}_i}^{-H_2(\mathrm{sk}_{\mathrm{id}_i} c_{\mathrm{is}})} \cdot H_1\left(\mathrm{pk}_{\mathrm{id}_i}^{\prime \mathrm{sk}'_{\mathrm{id}_i}}\right)\right) \\ \quad = m \cdot \hat{e}(g^{\alpha}, \mathrm{pk}_{\mathrm{id}_i}^{\gamma} H_2(\mathrm{sk}_{\mathrm{id}_i} \parallel t) \cdot \hat{e}\left(g^{\gamma}, \mathrm{sk}_{\mathrm{id}_i}^{-H_2(\mathrm{sk}_{\mathrm{id}_i} \| t)} \cdot H_1\left(\mathrm{pk}_{\mathrm{id}_i}^{\prime \mathrm{sk}'_{\mathrm{id}_i}}\right)\right) \\ \quad = m \cdot \hat{e}\left(g^{\gamma}, \left(\mathrm{pk}_{\mathrm{id}_i}^{\prime \mathrm{sk}'_{\mathrm{id}_i}}\right)\right) \\ c_{j3} = \mathrm{pk}'_{\mathrm{id}_i} \end{cases} \tag{7}$$

(8) $\mathrm{Dec}_{\mathrm{PKE}}(c_j, \mathrm{sk}'_{\mathrm{id}_j}, \mathrm{params})$: The decryption function of the ciphertext is re-encrypted. After receiving the ciphertext $c_j = (c_{j1}, c_{j2}, c_{j3})$ sent by the proxy server, the proxy receiver uses the private key $\mathrm{sk}'_{\mathrm{id}_i}$ of a traditional cryptography system to calculate the data plaintext using the following algorithm:

$$m' = \frac{c_{j2}}{\hat{e}(c_{j1}, H_1(\mathrm{pk}_{\mathrm{id}_i}^{\prime \mathrm{sk}'_{\mathrm{id}_j}}))} = \frac{m \cdot \hat{e}(g^{\gamma}, H_1(\mathrm{pk}'_{\mathrm{id}_i} \mathrm{sk}'_{\mathrm{id}_j}))}{\hat{e}(g^{\gamma}, H_1)(\mathrm{pk}_{\mathrm{id}_i}^{\prime \mathrm{sk}'_{\mathrm{id}_j}})} = m \tag{8}$$

(9) $\mathrm{ChaType}(C_{\mathrm{ID}}, t')$: This function is implemented when the data owner modifies the type of a file in order to change the authorization state or revoke authorization. The original ciphertext $c = (c_1, c_2, c_3)$ and the new fine-grained control type $t'$ is used to calculate the new ciphertext $c' = (c'_1, c'_2, c'_3)$ using the following steps:

$$\begin{cases} c'_1 = c_1, \\ c'_2 = c_2(\hat{e}(\mathrm{pk}_{\mathrm{id}}, \mathrm{pk})^{r \cdot H_2(\mathrm{sk}_{\mathrm{id}} \| t)})^{\frac{t'}{t}} \\ c'_3 = c_3 \end{cases} \tag{9}$$

The transformation from $c_2$ to $c'_2$ can also be obtained using $c'_2 = m\frac{t}{t'} \cdot c_2\frac{t'}{t}$.

## 4 Implementation

In this section, we introduce the system modules and functions in detail, and describe their characteristics and optimization in cloud environments. We use Cassandra as our distributed network database.

### 4.1 Overall architecture

The architecture of the improved PRE module is shown in Fig. 5. The user encrypts the data using key encapsulation via the client, by taking the following steps: (1) using AES to encrypt user data, (2) using the proxy server to re-encrypt the symmetric key for the encrypted file, and (3) storing the data from steps (1) and (2) together as the original ciphertext in the data center cluster.

Before the data owner stores data in the cloud, the file must first be encrypted with a symmetric encryption key (SEK). We call this the ciphertext body. The SEK is used with the private key in the proposed improved PRE scheme, to form the ciphertext of the SEK, SEKey, and the body and the SEKey are then stored together in the cloud. To share this file with recipients, the data owner computes re-encryption keys for trusted recipients. The proxy server is held by the cloud service provider and runs transparent proxy services. When the user sends a request to the cloud, the request distribution server sends this request to the proxy, which checks whether this user has the authority to access the requested file. If so, the proxy service obtains the re-encryption key from a database of re-encryption keys, and fetches the data, including the body and SEKey, from the data center. The proxy server then re-encrypts the SEKey to a type of ciphertext that the

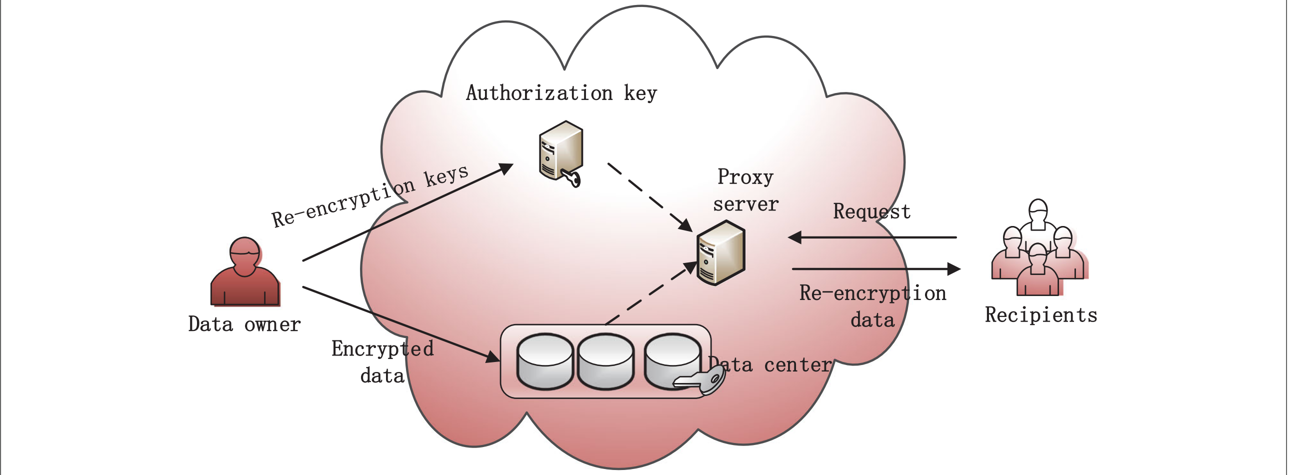

**Fig. 5** Architecture of PRE for data protection

recipient can decrypt with his own secret key. All of these processes taking place on the proxy server are transparent to the user. In this way, the user receives the required data files but they do not leave the proxy. When the recipient has gained access to the required files, he or she can decrypt the data using the client software.

In order to achieve the purpose of the fine-grained control, we add an element named "type" to the algorithm. Users can add a file to a type or many files to a single type. The type is included in the ciphertext when a data file is encrypted, and each re-encryption key for authorization is also of the same type, since the proxy cannot re-encrypt the data file successfully using a re-encryption key and data that do not match.

In addition, if after recipient has finished processing the data file need feedback to the user, and the data will then be used by a user with a different type, this can be achieved directly by computing $c' = (c'_1, c'_2, c'_3)$ , where

$c'_1 = c_1, c'_2 = m^{t \cdot t'^{-1}} \cdot c_2^{t' \cdot t^{-1}}, c'_3 = c_3$

This means that we can sidestep the process of encryption using the recipient's public key and sharing with the user, followed by decryption and sharing again. Any more, if user wants to recover authority or wants to change the type, this can be conveniently achieved by computing $c' = (c'_1, c'_2, c'_3)$ , where

$c'_1 = c_1, c'_2 = c_2 \cdot (e(\mathrm{pk}_{\mathrm{id}}, \mathrm{pk})^{r \cdot H_2(\mathrm{sk}_{\mathrm{id}} \| t)^{t' \cdot t^{-1}}}), c'_3 = c_3$

### 4.2 System processing flow

Key management is always an important issue in system implementation. In our system, we implement a KGC to generate and distribute key pairs. The user's IBE-type public key is the email address that was given when registering, and the IBE-type secret key is generated by KGC and held by the users themselves. The user's PKE-type secret key and public key are generated and distributed by CA. The search feature in our system was implemented by using the Cassandra NoSQL database, in which we can search for a data value by key. We also implement the user authorization function and friend management scheme in order to allow sharing of data files.The process used in our scheme is illustrated in Fig. 6.

The process nouns in Fig. 6 are explained in the following.

$\mathrm{body} = \mathrm{symmetricEnc}(M, \mathrm{SEK})$
$\mathrm{SEKey} = \mathrm{Enc}_{\mathrm{ibe}}(\mathrm{id}, \mathrm{params}, \mathrm{SEK}.t)$
$\mathrm{rk}_{(\mathrm{id}_i \to \mathrm{id}_k)} = \mathrm{KeyGen}_{\mathrm{pre}}(\mathrm{sk}_{(\mathrm{id}_i)}, \mathrm{sk}'_{\mathrm{id}_i}, \mathrm{pk}'_{(\mathrm{id}_j)}, t, \mathrm{params})$
$\mathrm{REQ} = \mathrm{FileAccessREQ}(\mathrm{fileKey}, \mathrm{userName})$
$C = \mathrm{body}, \mathrm{SEKey} = \mathrm{getTuple}(\mathrm{REQ})$
$\mathrm{SEKey}_j = \mathrm{ReEnc}(C_i, \mathrm{rk}_{(\mathrm{id}_i \to \mathrm{id}_j)}, \mathrm{params})$
$\mathrm{SEK} = \mathrm{Dec}_{\mathrm{pke}}(\mathrm{SEKey}_j, \mathrm{sk}'_{\mathrm{id}_i}, \mathrm{params})$
$M = \mathrm{symmetricDec}(M, \mathrm{SEK})$
$\mathrm{TCkey} = e(\mathrm{pk}_{\mathrm{id}}, \mathrm{pk})^{r \cdot H_2(\mathrm{sk}_{\mathrm{id}} \| t) t' \cdot t^{-1}}$
$\mathrm{REQT} = \mathrm{TypeREQ}(\mathrm{userName}, \mathrm{FileList}, \mathrm{TCkey})$
$C'_i = \mathrm{chaType}(C_i, t')$

## 5 Security analysis

**Theorem 1** *If the initial ciphertext and private key are correctly generated and meet the following conditions:* $C \leftarrow \mathrm{Enc}_{\mathrm{IBPRE}}(\mathrm{PP}_{\mathrm{IBPRE}}, U_{\mathrm{RoleId}}, m, \alpha)$, $\mathrm{SK}_{\mathrm{IBPRE}}^{\mathrm{UserId}} \leftarrow \mathrm{Extract}_{\mathrm{IBPRE}}(\mathrm{MK}_{\mathrm{IBPRE}}, \mathrm{UserId})$, *and* $\mathrm{UserId} \in U_{\mathrm{RoleId}}$. *Then, the plaintext m can be calculated by executing the algorithm* $\mathrm{Dec}_{\mathrm{IBPRE}}(\mathrm{PP}_{\mathrm{IBPRE}}, \mathrm{UserId}, \mathrm{SK}_{\mathrm{IBPRE}}^{\mathrm{UserId}}, C, U_{\mathrm{RoleId}})$.

*Proof* if $\mathrm{UserId} \in U_{\mathrm{RoleId}}$, then we have

$$\left(e(c_1, h^{(\Delta_\gamma, U_{\mathrm{RoleId}})}) \cdot e(\mathrm{SK}_{\mathrm{IBPRE}}^{\mathrm{UserId}}, c_2)\right)^{\frac{1}{\prod_{j=1, j \neq i}^{n} H_1(\mathrm{UserId}_j)}}$$
$$= \qquad (e(g^{-k \cdot \gamma}, h^{\Delta\gamma(\mathrm{UserId}, \mathrm{U}_{\mathrm{RoleId}})}) \qquad .$$

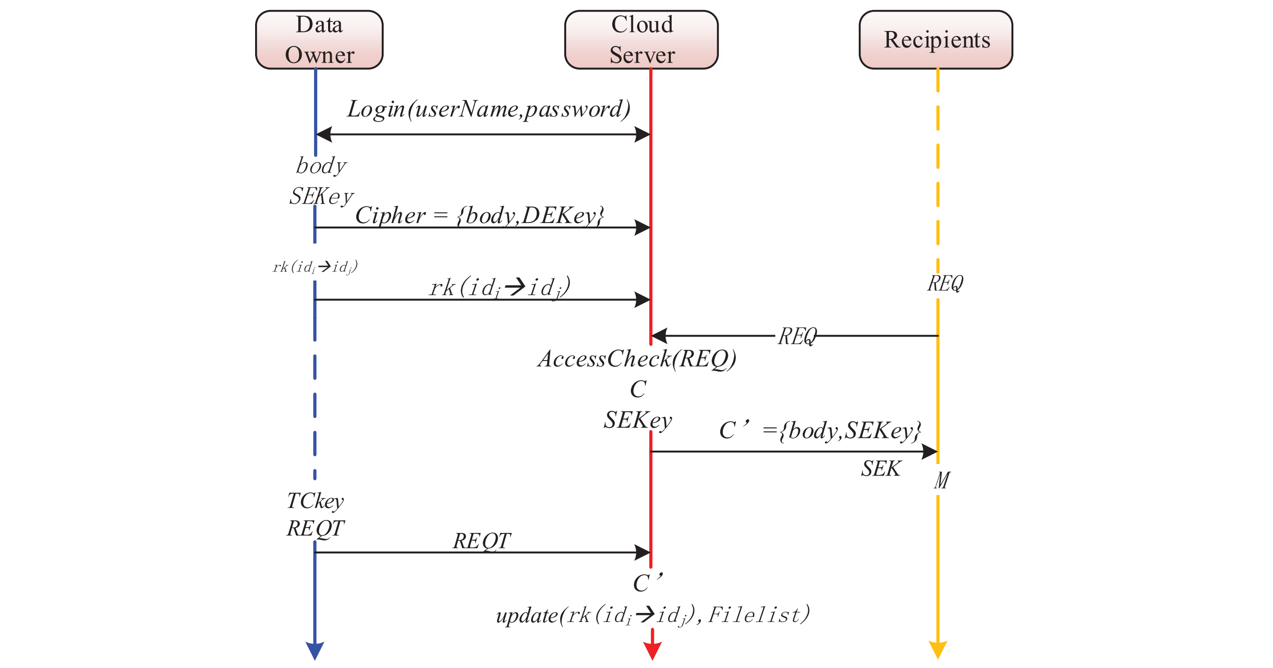

**Fig. 6** The system processing flow

$$e(g^{\frac{1}{\gamma+H_1(\text{userId})}}, h^{k\cdot\prod_{j=1}^{n}\gamma+H_1(\text{UserId}_j)}))^{\frac{1}{\prod_{j=1,j\neq i}^{n} H_1(\text{UserId}_j)}}$$
$$= (e(g^{-k\cdot\gamma}, h^{\Delta\gamma(\text{UserId},\text{U}_{\text{RoleId}})}) \cdot$$
$$e(g,h)^{k\cdot\prod_{j=1}^{n}\gamma+H_1(\text{UserId}_j)})^{\frac{1}{\prod_{j=1,j\neq i}^{n} H_1(\text{UserId}_j)}}$$
$$= (e(g,h)^{k\cdot\prod_{j=1}^{n}H_1(\text{UserId}_j)})^{\frac{1}{\prod_{j=1,j\neq i}^{n} H_1(\text{UserId}_j)}}$$
$$= v^k$$
$$m = \frac{c_3}{v^k}$$

Now, if UserId $\in U_{\text{RoleId}}$ ,then $\text{Dec}_{\text{IBPRE}}(\text{PP}_{\text{IBPRE}}, \text{UserId}, \text{SK}_{\text{IBPRE}}^{\text{UserId}}, U_{\text{RoleId}}) = m$. Similarly, the selected receiver can decrypt the correctly generated initial ciphertext into the original plaintext. □

**Theorem 2** *If the ciphertext is correctly generated by algorithm* $\text{ReEnc}_{\text{IBPRE}}()$ : $C' \leftarrow \text{ReEnc}_{\text{IBPRE}}(\text{PP}_{\text{IBPRE}}, d_{\text{UserId}\to U'_{\text{RoleId}|\alpha'}}, C, U_{\text{RoleId}})$*, and private key is correctly generated by algorithm* $\text{Extract}_{\text{IBPRE}}()$ : $SK_{\text{IBPRE}}^{\text{UserId}'} \leftarrow \text{Extract}_{\text{IBPRE}}(\text{MK}_{\text{IBPRE}}, \text{UserId}')$, $d_{\text{UserId}\to U'_{\text{RoleId}|\alpha'}} \leftarrow \text{RKExtract}_{\text{IBPRE}}(\text{PP}_{\text{IBPRE}}, \text{UserId}, \text{SK}_{\text{IBPRE}}^{\text{UserId}}, U'_{\text{RoleId}}, \alpha')$, $C \leftarrow \text{Enc}_{\text{IBPRE}}(\text{PP}_{\text{IBPRE}}, U_{\text{RoleId}}, m, \alpha)$*, and* $\text{SK}_{\text{IBPRE}}^{\text{UserId}} \leftarrow \text{Extract}_{\text{IBPRE}}(\text{MK}_{\text{IBPRE}}, \text{UserId})$ *are all executed correctly, and they are valid for any* UserId $\in U_{\text{RoleId}}, \alpha = \alpha'$*, and* UserId$' \in U'_{\text{RoleId}}$*. Then, execute* $\text{ReDec}_{\text{IBPRE}}(\text{PP}_{\text{IBPRE}}, \text{UserId}', \text{SK}_{\text{IBPRE}}^{\text{UserId}'}, C', U'_{\text{RoleId}})$*, thus we would obtain plaintext m.*

## 6 Simulation results and discussion

In this section, we evaluate the performance of the encryption and decryption by the data owner, re-encryption by the cloud proxy, and the performance of decryption by the recipients. These steps are implemented using the library of miracle on IBE and an elliptic curve defined on a 512-bit prime field with a generator order of 160 bits. The embedded degree of the curve is two. The experimental environment has the following configuration: a Ubuntu virtual machine (3.0 GHz vCPU, 1 GB RAM) running on a PC (Intel i5 3.0 GHz, 8 GB RAM).

### 6.1 Performance of encryption by data owner

The algorithm Enc(pk, aux,$M$) is executed on the client side. It does not reduce the use of bilinear mapping, and the calculation time is not significantly optimized.

Figure 7 shows the computational overhead of the data owner when performing different frequency encryption operations on the system. This step is executed when the data owner uploads a data file; the data must first be encrypted with a random 128-bit AES symmetric key, and this key is then encrypted using the encryption algorithm in our scheme. Our scheme consumes about the same amount of time as the standard PRE scheme; this step takes less than 20 ms, which is acceptable for a real system.

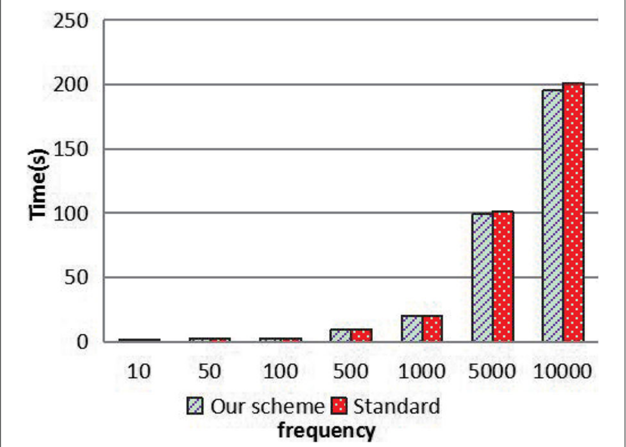

**Fig. 7** Performance of encryption by data owner

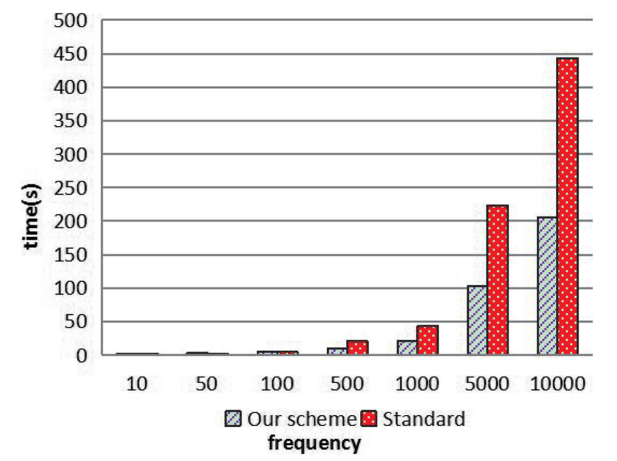

**Fig. 9** Performance of delegation by data owner

## 6.2 Performance of decryption by data owner

Decryption of the original ciphertext function is performed by the data owner via the client. The algorithm $\mathrm{Dec}_{\mathrm{PKE}}(\mathrm{sk}, \mathrm{aux}, C_{\mathrm{PK}})$ cannot reduce the number of bilinear mappings. Figure 8 shows the computational overhead of the data owner when performing different frequency decryption operations on our system. This step is executed when the data owner downloads and decrypts data. In this step, the ciphertext is not transformed by the proxy server. The proposed scheme consumes about the same amount of time as the standard PRE scheme; this step takes less than 15 ms, and since this is carried out only once per file, this is an acceptable value.

## 6.3 Performance of delegation by data owner

Figure 9 shows the computational overhead due to the data owner performing different frequency delegation operations on the system. This step is executed by the data owner after the encrypted data file is uploaded. The delegation progress involves generating a PRE key and sending it from the data owner to a recipient of the class file of a given type. The more recipients with whom the data owner wants to share files, the more times this sharing step is performed, meaning that this step has a significant impact on the overall client performance. As we can see from Fig. 8, our solution takes less than half the time required by the standard PRE solution; the proposed solution takes around 20 ms, while standard PRE takes around 45 ms, meaning that the optimization of this step is significant.

## 6.4 Performance of re-encryption by cloud proxy

Figure 10 shows the computational overhead of the cloud proxy as it performs different frequency re-encryption operations in our scheme. This step transforms the ciphertext from the data owner to the recipients. The time

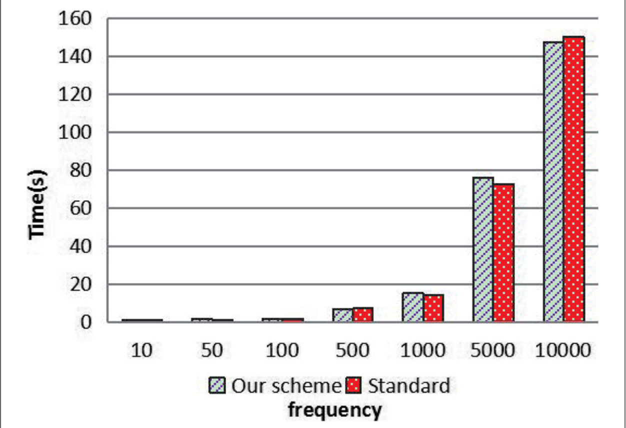

**Fig. 8** Performance of decryption by data owner

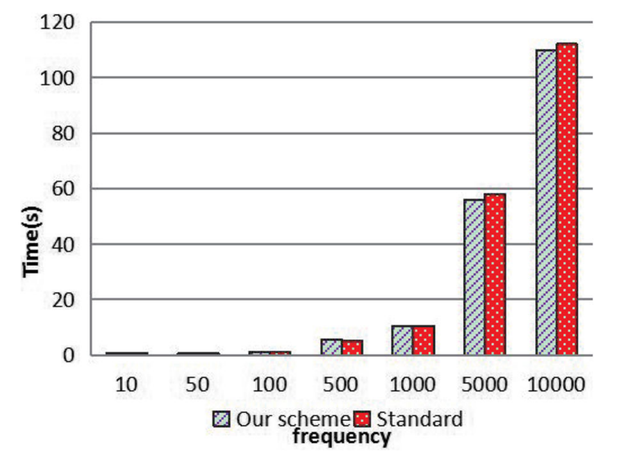

**Fig. 10** Performance of re-encryption

costs of our scheme in this process are also similar to those of standard PRE, with an average time of about 12 ms. As is well-known, the computing resources of cloud platforms are abundant, and the development of hardware is also taking place rapidly. Cloud service providers can therefore take many measures to speed up this process, such as dedicating more hardware resources or using visualization, meaning that the cost of this step would not be onerous to the cloud service provider.

### 6.5 Performance of decryption by recipient

The decryption algorithm is executed when the recipient decrypts the re-encrypted data shared by the data owner, and this algorithm is executed by the client. As can be seen from Fig. 11, our scheme is about 47% better than the standard PRE scheme. The average times required by our scheme and the standard PRE scheme are about 21 and 40 ms respectively, giving a very high proportion of optimization.

## 7 Conclusion

The aim of this paper is to develop an improved PRE scheme for protecting users' private data. Our scheme is unique in that it integrates fine-grained delegation based on the element of type and heterogeneous features that can transform ciphertext from IBE-type to PKE-type text. The fine-grained features mean that the data owner can share private data using a fine-grained approach, e.g., adding a single file or a class of files. The feature of heterogeneity greatly improves the performance of the algorithm and at the same time makes it more convenient and compatible with the system developed based on the traditional PKE encryption algorithm. An interesting direction for future work would be to make this scheme more secure by supporting other ciphertext security schemes and addressing other security issues. We also need to continue to optimize the performance and make this scheme more practical and to carry out research into dynamic data privacy protection in the cloud.

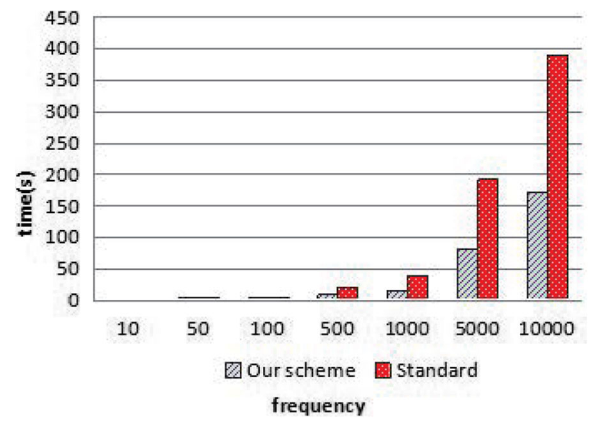

**Fig. 11** Performance for decryption by recipient

### Abbreviations

ABE: Attribute-based encryption; IBE: Identity-based encryption; PRE: Proxy re-encryption

### Acknowledgements

We thank Puming Wang for feedback on the initial versions of this paper.

### Authors' contributions

JNS contributed to the main idea and drafted the manuscript, algorithm design, and performance analysis. XJD and ZWX contributed to the simulations. All authors read and approved the final manuscript.

### Funding

The work was supported by the National Natural Science Foundation of China under grant no. 61662022.

### Availability of data and materials

Not applicable

### Competing interests

The authors declare that they have no competing interests.

Received: 10 September 2019 Accepted: 4 December 2019
Published online: 18 December 2019

### References

1. X. Wang, L. T. Yang, L. Kuang, X. Liu, Q. Zhang, M. J. Deen, A tensor-based big-data-driven routing recommendation approach for heterogeneous networks. IEEE Netw. **33**(1), 64–69 (2019)
2. X. Wang, L. T. Yang, X. Xia, J. Jin, M. J. Deen, A cloud-edge computing framework for cyber-physical-social services. IEEE Commun. Mag. **55**(11), 80–85 (2017)
3. P. Wang, L. T. Yang, J. Li, J. Chen, S. Hu, Data fusion in cyber-physical-social systems: state-of-the-art and perspectives. Inf Fusion. **51**, 42–57 (2019)
4. L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, J. Chen, A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service re commendation in cross-platform edge environment. Futur. Gener. Comput. Syst. **88**, 636–643 (2018). https://doi.org/10.1016/j.future.2018.02.050
5. W. Gong, L. Qi, Y. Xu, Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment. Wirel. Commun. Mob. Comput. **2018**(3075849), 1–8 (2018)
6. Q. GAN, X. WANG, X. FANG, Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud. Sci. China Inf. Sci. **61**(12), 97–111 (2018)
7. R. Li, C. Shen, H. He, X. Gu, Z. Xu, C.-Z. Xu, A lightweight secure data sharing scheme for mobile cloud computing. IEEE Trans. Cloud Comput. **6**(2), 344–357 (2017)
8. M. Sookhak, F. R. Yu, M. K. Khan, X. Yang, R. Buyya, Attribute-based data access control in mobile cloud computing: taxonomy and open issues. Futur. Gener. Comput. Syst. **72**(C), 273–287 (2017)
9. B. Liu, X. L. Yu, S. Chen, X. Xu, L. Zhu, in *2017 IEEE International Conference on Web Services (ICWS)*. Blockchain Based Data Integrity Service Framework for IoT Data (IEEE, 2017), pp. 468–475
10. L. Qi, Q. He, F. Chen, W. Dou, S. Wan, X. Zhang, X. Xu, Finding all you need: web apis recommendation in web of things through keywords search. IEEE Trans. Comput. Soc. Syst. **6**(5), 1–10 (2019)
11. L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, X. Xu, A qos-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. World Wide Web, 1–23 (2019). https://doi.org/10.1007/s11280-019-00684-y
12. X. Wang, L.T. Yang, X. Chen, M.J. Deen, J. Jin, Improved multi-order distributed hosvd with its incremental computing for smart city services. IEEE Trans. Sustain. Comput., 1–14 (2018). https://doi.org/10.1109/TSUSC.2018.2881439
13. X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, B. O. Apduhan, Nqa: A nested anti-collision algorithm for rfid systems. ACM Trans. Embed. Comput. Syst. (TECS). **18**(4), 1–21 (2019)

14. J. Zhou, J. Yan, K. Cao, Y. Tan, T. Wei, M. Chen, G. Zhang, X. Chen, S. Hu, Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous mpsocs. Inf. Fusion. **82**, 1–11 (2018)
15. M. B. Smithamol, S. Rajeswari, Hybrid solution for privacy-preserving access control for healthcare data. Adv. Electr. Comput. Eng. **17**(2), 31–38 (2017)
16. Y. He, J. Ni, X. Wang, B. Niu, F. Li, X. Shen, Privacy-preserving partner selection for ride-sharing services. IEEE Trans. Veh. Technol. **67**(7), 5994–6005 (2018)
17. H. Cui, R. H. Deng, J. K. Liu, Y. Li, in *Australasian Conference on Information Security & Privacy(ACISP 2017)*. Attribute-Based Encryption with Expressive and Authorized Keyword Search (Springer, Cham, 2017), pp. 106–126
18. M. Zhang, Y. Jiang, Y. Mu, W. Susilo, Obfuscating re-encryption algorithm with flexible and controllable multi-hop on untrusted outsourcing server. IEEE Access. **5**, 26419–26434 (2017)
19. L. Jiang, D. Guo, Dynamic encrypted data sharing scheme based on conditional proxy broadcast re-encryption for cloud storage. IEEE Access. **5**(99), 13336–13345 (2017)
20. C. Zuo, J. Shao, Z. Liu, Y. Ling, G. Wei, in *2017 IEEE Trustcom/BigDataSE/ICESS*. Hidden-Token Searchable Public-Key Encryption (IEEE, 2017), pp. 248–254
21. H. Li, D. Liu, Y. Dai, T. H. Luan, S. Yu, Personalized search over encrypted data with efficient and secure updates in mobile clouds. IEEE Trans. Emerg. Top. Comput. **6**(1), 97–109 (2015)
22. X. Ding, P. Liu, H. Jin, Privacy-preserving multi-keyword top-$k$ k similarity search over encrypted data. IEEE Trans. Dependable Secure Comput. **16**(2), 344–357 (2017)
23. Z. Shen, J. Shu, X. Wei, Preferred search over encrypted data. Frontiers of Computer Science. **12**(3), 593–607 (2018)
24. S. Xu, G. Yang, Y. Mu, A new revocable and re-delegable proxy signature and its application. J. Comput. Sci. Technol. **33**(2), 380–399 (2018)
25. J. Ni, K. Zhang, X. Lin, X. S. Shen, Securing fog computing for internet of things applications: challenges and solutions. IEEE Commun. Surv. Tutor. **20**(1), 601–628 (2017)
26. P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, H. Jin, Generating searchable public-key ciphertexts with hidden structures for fast keyword search. IEEE Trans. Inf. Forensic. Secur. **10**(9), 1993–2006 (2015)
27. M. Abe, J. Camenisch, R. Dowsley, M. Dubovitskaya, in *Theory of Cryptography Conference(TCC 2014)*. On the Impossibility of Structure-Preserving Deterministic Primitives (Springer, Berlin, 2014), pp. 713–738
28. Y. Guo, F. Liu, Z. Cai, N. Xiao, Z. Zhao, Edge-based efficient search over encrypted data mobile cloud storage. Sensors. **18**(4), 1–14 (2018)
29. M. Kim, H. T. Lee, S. Ling, H. Wang, On the efficiency of fhe-based private queries. IEEE Trans. Dependable Secur. Comput. **15**(2), 357–363 (2016)
30. P. Gasti, J. Sedenka, Q. Yang, Z. Gang, K. S. Balagani, Secure, fast, and energy-efficient outsourced authentication for smartphones. IEEE Trans. Inf. Forensic. Secur. **11**(11), 2556–2571 (2016)
31. F. Tramer, F. Zhang, H. Lin, J.-P. Hubaux, A. Juels, E. Shi, in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. Sealed-glass proofs: using transparent enclaves to prove and sell knowledge (IEEE, 2017), pp. 19–34

## Publisher's Note