Hierarchical Access Control with Scalable Data Sharing in Cloud Storage

Zhenyao Qiu[1], Zhiwei Zhang[1], Shichong Tan[1], Jianfeng Wang[1], Xiaoling Tao[2,3]

[1] State Key Laboratory of Integrated Services Networks (ISN), Xidian University, China
[2] Guangxi Cooperative Innovation Center of Cloud Computing and Big Data, Guilin University of Electronic Technology, China
[3] Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, China
zyqiu@stu.xidian.edu.cn, zwzhang@xidian.edu.cn, sctan@mail.xidian.edu.cn, jfwang@xidian.edu.cn, txl@guet.edu.cn

Abstract

Cloud storage is facing the contradiction between data security and flexible data sharing, and therefore the cryptographic access control mechanisms are well studied. In particular, hierarchical access control in cloud storage is significant for many application scenarios. In these scenarios, the users are divided into several groups organized in a hierarchy, and they are assigned with different access privileges according to their groups and levels. That is, the users in higher level groups can access the data belonging to their subordinate groups while the users in lower level groups cannot access the data belonging to their superior groups. However, most of the existing hierarchical access control solutions seem to be unpractical for their inability of scalable data sharing, inefficiency of key management or lack of delegated re-encryption. In this paper, we propose a new hierarchical access control scheme based on key-aggregate encryption, and the proposed scheme realizes scalable data sharing in cloud storage which allows the users to share data with any user group. In the proposed scheme, the size of each key or ciphertext is constant and irrelevant to the scale of hierarchical user structure. Especially, our scheme improves the convenience of key management by cutting off the key derivation widely used in the existing hierarchical key assignment methods. Furthermore, the proposed scheme reduces the users' updating overhead by introducing the delegated re-encryption into the hierarchical scenarios. Finally, the security analysis and the performance evaluation indicate that our scheme is feasible for the hierarchical data sharing applications in cloud storage.

Keywords: Access control, Hierarchical key assignment, Data sharing, Cloud storage, Key-aggregate encryption

1 Introduction

Cloud computing has achieved the dream of computing being the 5th utility after water, electricity, gas, and telephony [1]. Based on elastic virtualization technology, computational services can be provided to resource-limited users via the Internet on demand. Plenty of technologies in cloud computing have been well studied these years, such as secure outsourcing computation [2-3]. With the development and maturity of cloud computing technologies, cloud storage service, a typical representative of cloud computing services, has attracted increasingly more organizations and personal users over last ten years. Compared with traditional data storage methods in which users should purchase and maintain physical devices and software by themselves, cloud storage has a lot of advantages including lower cost, on-demand selection, ease of management and anywhere accessibility. Nowadays, many information technology companies have launched cloud storage services such as Google Drive, Microsoft OneDrive, Amazon S3, etc.

However, users' data may contain sensitive information, such as government confidential documents, enterprise business data, and personal medical records. Although cloud storage may bring conveniences to users, it will also bring data security threats to them at the same time. As a result, the security issues of cloud storage have received attentions from both academia and industry, including data auditing [4-7], secure data deduplication [8-9] and searchable encryption [10-11]. On one hand, the adoption of outsourcing storage will cause the separation of data ownership and management. Once the data has been uploaded to the cloud server, it can only be accessed and processed by users in a remote way. That is to say, users lost the autonomy of their data. As the real manager of users' data, the cloud

*Corresponding Author: Zhiwei Zhang; E-mail: zwzhang@xidian.edu.cn
DOI: 10.3966/160792642019052003002

service providers are often considered to be semi-trusted which means that they cannot be fully trusted by users since they may steal or even divulge users' data stored in the cloud. Therefore, it is necessary for cloud users to utilize proper cryptographic mechanisms before delivering their data to cloud servers in order to protect data security. On the other hand, the data stored in the cloud sometimes needs to be accessed by different users within certain access control policies. The existing data access control solutions in cloud storage are generally constructed with attribute-based encryption (ABE) [12-17]. For instance, in ciphertext-policy attribute-based encryption (CP-ABE) [14], a user will be assigned with a set of specific attributes according to his/her identity, while the files with different access permission requirements will be attached with different access control policies. Once a user's attributes meet a file's access control policy, he/she can decrypt the file.

Secure data sharing is also an important functionality in cloud storage. At the first glance, the data owner can share data by first decrypting the data downloaded from the cloud server and then sending the plaintext to others. However, a more ideal and secure situation is that the users should be given proper decryption rights and can directly decrypt the shared data in the cloud server if they satisfy the conditions of sharing targets made by the data owner.

Nevertheless, in some specific real-world application scenarios of cloud storage, such as enterprise cloud and government cloud, data users are divided into several groups in a hierarchy with hierarchical access privileges. Users in a higher-level group have the authority to access the data of their subordinate groups, while users in a lower-level group cannot access the data belonging to their superior groups. Under this requirement, the original ABE is infeasible to be directly used to solve the access control problem in such scenarios. Consequently, the hierarchical attribute-based encryption (HABE) is proposed to deal with the access control problems in hierarchical scenarios. However, the existing HABE schemes mainly consider the hierarchical structure of attribute authorities but not the hierarchical structure of user groups.

For example, as shown in Figure 1, in an application scenario of enterprise cloud storage, the ordinary staffs (e.g. engineers) of each department can only access the data of their own groups, department managers (e.g. managers of R&D department) can access the data of their groups and all their subordinate groups, and the enterprise leadership (e.g. CEO) has the supreme seniority and can access all data of the whole company [18]. Another important requirement for hierarchical access control is flexible data sharing. For instance, an auditor may share some information about the misconduct of the managers of the financial department with CEO but wants to keep secret to the managers. In conclusion, under the condition of ensuring the data confidentiality in cloud storage, it is of great significance to solve the problem of secure data sharing and access control when users are organized in a hierarchical structure.

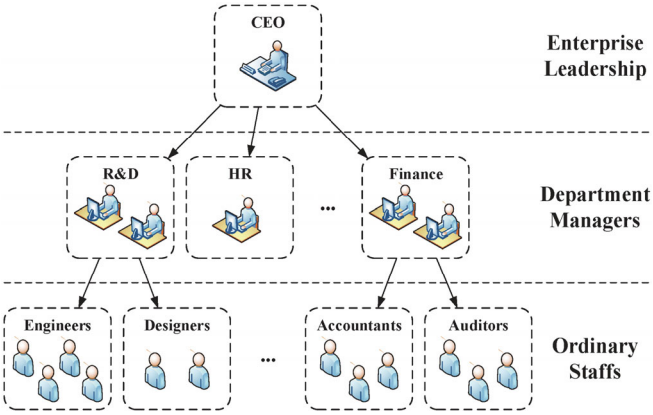


Figure 1. An example of hierarchical sccess control

1.1 Our Contribution

The main contributions of this paper are listed below:

- We achieve a hierarchical access control scheme under the public-key cryptosystem based on the key-aggregation encryption. In our scheme, each user only needs to be distributed with a single secret key of constant size to decrypt the data belonging to his/her group and all its subordinate groups.
- The users in our scheme can share encrypted data with any of the other user groups without using others' secret keys, which is suitable for the scenario of secure data sharing in cloud storage. Unlike many existing hierarchical key assignment schemes, the decryption process of our scheme does not require a key derivation step. A user can directly use the secret key of his/her own group to decrypt the data of all its subordinate groups.
- Our scheme also supports delegated re-encryption. That is to say, in the process of user revocation, the cloud server is delegated to re-encrypt the data belonging to the user group which contains the revoked user and its subordinate groups.

1.2 Related Work

Akl and Taylor [19] first proposed the problem of hierarchical access control and presented a cryptographic solution. They described the hierarchical structure of user groups as the partial order relations among security classes, and presented a collusion-resistant hierarchical key assignment scheme based on some theorems in number theory. However, the key size in this scheme will increase rapidly with the increasing of breadth and depth of the hierarchical user structure. Then, Sandhu [20] proposed an implementation of hierarchical access control in a tree structure based on the idea of one-way function to keep the key size of

each user constant. Damiani et al. [21] presented an access control scheme of outsourced databases for multi-user in a tree hierarchy using the key derivation method based on one-way function proposed in [20]. Atallah et al. [22] proposed a dynamic hierarchical key assignment scheme which represents the partial order relations among security classes as a directed acyclic graph (DAG) and gives a specific public information for each vertex and edge in the graph. The scheme utilizes hash function to achieve the key derivation from any security class to its descendant classes and the dynamic updates of the hierarchy only occur locally. Later, Crampton et al. [23] analyzed and compared these hierarchical access control schemes and defined a general model of hierarchical key assignment.

In recent years, Chen et al. [24-25] constructed two hierarchical access control schemes under Bell-LaPadula security model based on proxy re-encryption and all-or-nothing transformation, identity-based broadcast encryption and strong one-time signature scheme, respectively. Castiglione et al. [26-28] proposed several solutions for different application scenarios. They presented a new hierarchical and shared access control model and constructed two concrete schemes [26] based on symmetric encryption and public-key threshold broadcast encryption, and then they achieved a new scheme [28] supporting dynamic updates in cloud storage based on Akl-Taylor scheme [19]. Tang et al. [18] proposed a solution of hierarchical key assignment with efficient key derivation based on linear geometry. Alderman et al. [29] proposed a tree-based cryptographic access control scheme mapping the access policy into a binary tree and achieving key derivation without public system information. Zhang et al. [30] proposed a hierarchical verifiable database scheme supporting partial verification and tampered record location based on a new cryptographic primitive called vector commitment tree, which beyond the verifiable database schemes [31-32] based on vector commitments [33] in both function and efficiency.

Based on ABE, Wang et al. [34] first proposed HABE for fine-grained access control in cloud storage by combining hierarchical identity-based encryption and CP-ABE. The HABE scheme achieves scalable user revocation via proxy re-encryption and lazy re-encryption. Wan et al. [35] proposed the hierarchical attribute-set-based encryption by extending ciphertext-policy attribute-set-based encryption with a hierarchical structure of users. Huang et al. [36] presented a data collaboration scheme with hierarchical attribute-based encryption in cloud computing based on HABE and attribute-based signature.

In the field of secure data sharing in cloud computing, there are some new schemes proposed recently. Shen et al. [37-38] first proposed a block design-based key agreement protocol for group data sharing based on symmetric balanced incomplete block design (SBIBD), and then proposed a traceable group data sharing scheme with traceability and anonymity utilizing SBIBD. Li et al. [39] presented a secure attribute-based data sharing scheme for resource-limited users based on ABE. Zhang et al. [40] proposed a location-sensitive secure data sharing scheme for cyber-physical systems with cloud computing.

Most of the existing hierarchical access control solutions are under the symmetric cryptosystem which do not well solve the problem of scalable and secure data sharing in cloud storage, while HABE-based schemes are costly in computational overhead and are not suitable for the scenarios of hierarchical user groups.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we introduce some preliminaries used in our concrete scheme. In Section 3, we present the system description and security model of our scheme. The framework and concrete construction of our hierarchical access control scheme are provided in Section 4. Then, Section 5 and Section 6 present the security analysis and performance analysis of the scheme respectively. Finally, we conclude this paper in Section 7.

2 Preliminaries

In this section, we first briefly introduce the bilinear pairing, which is a main building block of our concrete scheme. Then, we introduce the framework of key-aggregate cryptosystem, on which our proposed scheme is based.

2.1 Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. A bilinear pairing $\hat{e}:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ is a map with the following properties:

- *Bilinear*: $\forall g_1,g_2\in\mathbb{G}$ and $\forall a,b\in\mathbb{Z}$, we have $\hat{e}(g_1^a,g_2^b)=\hat{e}(g_1,g_2)^{ab}$.
- *Non-degenerate*: $\hat{e}(g,g)\neq 1$.
- *Computable*: $\forall g_1,g_2\in\mathbb{G}$, there is an efficient algorithm to compute $\hat{e}(g_1,g_2)$.

The group $\mathbb{G}$ is called a bilinear group. Many classes of elliptic curves feature bilinear groups.

2.2 Key-Aggregate Encryption

The key-aggregate cryptosystem (KAC) is proposed in [41] to address the flexible one-to-one data sharing problem in cloud storage. It allows a data owner to choose any set of different data classes and share them with a user (delegatee) by aggregating the decryption

power of different ciphertext classes into a single key for the delegatee to obtain the original data in the pre-chosen data set.

A KAC scheme consists of five polynomial-time algorithms as follows.

The data owner establishes the public system parameter via **Setup** and generates a public/master-secret key pair via **KeyGen** . Messages can be encrypted via **Encrypt** by anyone who also decides what ciphertext class is associated with the plaintext message to be encrypted. The data owner can use the master-secret key to generate an aggregate decryption key for a set of ciphertext classes via **Extract** . The generated keys can be passed to delegatees securely (via secure e-mails or secure devices). Finally, any user with an aggregate key can decrypt any ciphertext provided that the ciphertext's class is contained in the aggregate key via **Decrypt** .

- **Setup**$(1^{\lambda}, n)$: executed by the data owner to setup an account on an *untrusted* server. On input a security level parameter 1^{λ} and the number of ciphertext classes n (i.e., the class index should be an integer bounded by 1 and n), it outputs the public system parameter $param$, which is omitted from the input of the other algorithms for brevity.
- **KeyGen** : executed by the data owner to randomly generate a public/master-secret key pair (pk, msk) .
- **Encrypt**(pk, i, m) : executed by anyone who wants to encrypt data. On input a public-key pk , an index i denoting the ciphertext class, and a message m , it outputs a ciphertext $\mathcal{C}$.
- **Extract**$(msk, \mathcal{S})$: executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegatee. On input the master-secret key msk and a set $\mathcal{S}$ of indices corresponding to different classes, it outputs the aggregate key for set $\mathcal{S}$ denoted by $K_{\mathcal{S}}$.
- **Decrypt**$(K_{\mathcal{S}}, \mathcal{S}, i, \mathcal{C})$: executed by a delegatee who received an aggregate key $K_{\mathcal{S}}$ generated by **Extract**. On input $K_{\mathcal{S}}$, the set $\mathcal{S}$, an index i denoting the ciphertext class the ciphertext $\mathcal{C}$ belongs to, and $\mathcal{C}$, it outputs the decrypted result m if $i \in \mathcal{S}$.

With the inspiration of KAC, we construct a cryptographic access control scheme to fit the scenario of users with a hierarchical structure.

3 Models and Assumptions

In this section, we first describe the system model and problem statement for our scheme. Then we give the security assumptions and definitions in the scenario of hierarchical access control in cloud computing.

3.1 System Description

The system model for our proposed scheme consists of three parties: *Central Authority*, *Cloud Server* and *Hierarchical User Structure* described as follows. Figure 2 shows a simple framework of our system model.

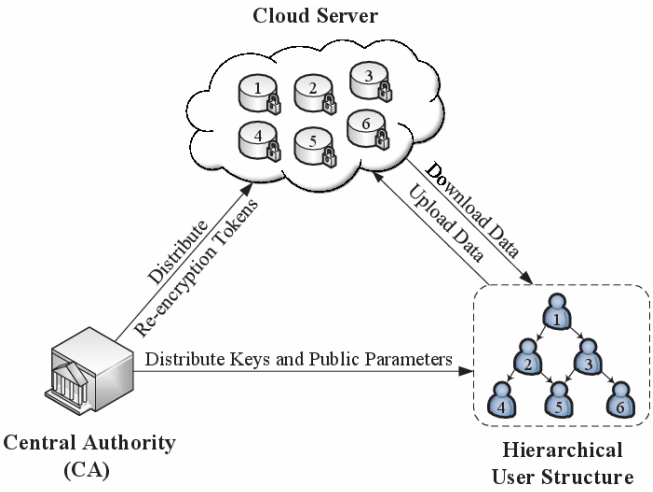


Figure 2. System model

- *Central Authority*: Central Authority (CA) is a trusted party for both cloud server and hierarchical user structure whose responsibilities are generating system parameters, distributing keys and public parameters to each user and providing some materials for the cloud server to update the data when needed.
- *Cloud Server*: The cloud server is a virtualized storage service provided by a cloud service provider. It is responsible for storing data, handling upload/download requests and updating data. It is said to be a semi-trusted one in our proposed scheme.
- *Hierarchical User Structure*: The hierarchical user structure consists of several user groups organized in a hierarchy and each group has one or more user(s) with the same level of access privilege. The users can send upload/download requests to the cloud server for the purpose of data sharing or data accessing. A user can share data with any user group in the structure, but can only access the data belonging to his/her group or its subordinate groups. That is to say, he/she cannot access any datum of his/her superior groups.

The hierarchical structure of user groups can be represented as a partial order set $(V, \preceq)$, where $V = \{SC_1, \cdots, SC_n\}$ is the set of user groups. The element SC_i represents a single user or an access group which consists of several users with the same access privilege. The binary relation $\preceq$ represents the hierarchical relations among the elements in V . For example, $SC_i \preceq SC_j$ means that the users in group

SC_j have the permission to access the data belonging to group SC_i. That is to say, the access priority of group SC_j is higher than or equal to that of group SC_i. If $SC_i \preceq SC_j$ and $SC_i \neq SC_j$, then $SC_i \prec SC_j$.

In the view of graph theory, any partial order set $(V, \preceq)$ can be represented as a DAG $G=(V,E)$. If $SC_i, SC_j \in V$ and $SC_i \prec SC_j$, there exists an edge from SC_j to SC_i in E. As shown in Figure 3, we can see that $SC_2, SC_4, SC_5 \preceq SC_2$ but $SC_1, SC_3, SC_6 \not\preceq SC_2$. In this paper, we use DAG $G=(V,E)$ as the parameter describing the hierarchical structure of user groups.

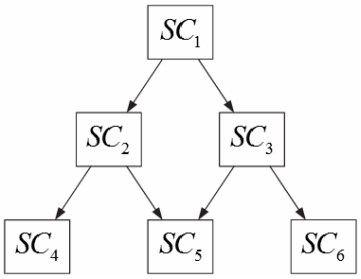


Figure 3. A DAG with 6 vertices and 6 edges

3.2 Security Model

In our scheme, we consider the cloud server as a semi-trusted one or in other words an honest-but-curious one. That is to say, the cloud server will execute every operation honestly in general, but can be interested in viewing user's content and try to find out as much secret information as possible based on their inputs. We assume that the cloud server would not collude with malicious users including someone whose access privileges were revoked. The communication channel between CA and both users and the cloud server are assumed to be secured under existing security protocols such as SSL. Users would try to access files either within or beyond the scope of their access privileges. To achieve this goal, unauthorized users may work independently or cooperatively.

The following security requirements should be guaranteed in our system.

- *Data Confidentiality*: A user or external attacker cannot access the data belonging to users with higher access privilege using his/her own group key. That is to say, users in group i cannot decrypt the data belonging to group j if $SC_j \not\preceq SC_i$.
- *Collusion Resistant*: Multiple malicious users (including external attackers) cannot collude to obtain the data belonging to a user with hierarchical access privilege beyond these colluded users.
- *Forward Security*: A user u cannot access the future uploaded data of his/her past group i and all its descendant groups $\{j \mid SC_j \preceq SC_i\}$ after u's access privilege is revoked from SC_i.

4 Our Proposed Scheme

In this section, we first introduce the overview of our hierarchical access control scheme with data sharing. Then, we present a concrete construction of the scheme. Finally, we give some advice in handling dynamic updates of hierarchical user structure.

Table 1 lists the main symbols used in our scheme.

Table 1. Symbols and their meanings

Symbols	Meanings
G	Graph of hierarchical user structure
n	Total number of user groups in G
SC_i	User group with index i
$\mathcal{S}_i$	Indices set of SC_i's subordinate groups
msk	System master secret key
γ_i	System secret parameter of user group i
$paramf$	Public parameters
M	Public matrix
dk_i	Private decryption key of user group i
ek_i	Public encryption key of user group i
$\mathcal{C}_m$	Ciphertext of message m
m_d	Decrypted plaintext of ciphertext $\mathcal{C}_m$
rk_i	Re-encryption token of user group i

4.1 Framework

Our scheme is inspired by [41] which uses the collusion-resistant broadcast encryption scheme proposed by [42] as the building block. The main procedures and functions contained in our scheme are described as follows.

System Initialization: CA runs the system setup algorithm **Setup**. On input the security parameter 1^λ and the DAG G representing the hierarchical structure of user groups, **Setup** algorithm outputs the system master secret key msk, the system parameters $\{\gamma_i\}$ representing n user groups, a public matrix M and the system public parameter $param$. CA makes M and $param$ public, and keeps msk and $\{\gamma_i\}$ secret.

Hierarchical Key Assignment: CA runs the key generation algorithm **KeyGen** to generate private decryption keys and public encryption keys for all user groups. On input the hierarchical structure G, system public parameter $param$, index j of the group to be distributed key to and the system parameters $\{\gamma_i\}$ representing n user groups, **KeyGen** algorithm outputs the private decryption key dk_j and the public encryption key ek_j for group j. CA distributes dk_j to

all users in group j and makes ek_j public.

Data Sharing: A user can choose any group to share data with when needed, including his/her own group, one of his/her subordinate groups, one of his/her superior groups or any other group. Once an encrypted file is uploaded, only the users in the target group and its superior groups can decrypt the file.

Once a user wants to share a file *DataFile* to a target group k, he/she can first choose a symmetric encryption algorithm, randomly generate a symmetric encryption key *DEK*, and encrypt the file into $\{DataFile\}_{DEK}$ with the chosen symmetric encryption algorithm, then she can run the data encryption algorithm **Encrypt** to generate the ciphertext header. On input the system public parameter *param*, the symmetric encryption key *DEK* of the file to be shared, the index k of target group and its public encryption key ek_k, **Encrypt** algorithm outputs the ciphertext header $\mathcal{C}_{DEK}$. The user uploads the ciphertext header $\langle k, \mathcal{C}_{DEK} \rangle$ with the ciphertext body $\{DataFile\}_{DEK}$ to the cloud server.

Data Decryption: Any user can download the ciphertext header $\langle k, \mathcal{C}_{DEK} \rangle$ with the ciphertext body $\{DataFile\}_{DEK}$, and run the data decryption algorithm **Decrypt** to attempt to decrypt the data. On input the hierarchical structure G, public matrix M, system public parameter *param*, the index j of the group where the user is, the corresponding private decryption key dk_j and the ciphertext header $\langle k, \mathcal{C}_{DEK} \rangle$, **Decrypt** algorithm outputs the symmetric encryption key *DEK* of the file if group j is a subordinate group of group k. The user can then decrypt the ciphertext body $\{DataFile\}_{DEK}$ with *DEK* to obtain the original file *DataFile*.

User Revocation: Once a user in group l needs to be removed from the group, to ensure the forward security, his/her access permissions to group l and its subordinate groups should be revoked from the system.

Firstly, CA runs the key update algorithm **Update** to update the private decryption keys of group l where the revoked user is and other related groups. On input the hierarchical structure G, the system master secret key *msk*, the system parameters $\{\gamma_i\}$ representing n user groups, the system public parameter *param* and the index l of the group where the revoked user is, **Update** algorithm outputs an updated public matrix M', new secret parameter $\{\gamma_{q'}\}$, new public encryption keys $\{ek_{q'}\}$ and re-encryption tokens $\{rk_q\}$ of group l and its subordinate groups $\{q \mid SC_q \preceq SC_l\}$, as well as new private decryption keys $\{dk_{k'}\}$ of groups $\{q \mid SC_q \preceq SC_l\}$ and their superior groups $\{k\} = \{q\} \cup \{i \mid SC_i \succeq SC_q\}$. CA makes M' and $\{ek_{q'}\}$ public, and distributes $\{dk_{k'}\}$ to all users in groups $\{k\}$. CA sends the re-encryption tokens $\{\langle q, rk_q \rangle\}$ to the cloud server.

Then, the cloud server runs the data re-encryption algorithm **ReEncrypt** for each involved group in group l and its subordinate groups. On input the ciphertext header $\langle q, \mathcal{C}_{DEK} \rangle$ and the corresponding re-encryption token rk_q, **ReEncrypt** algorithm outputs the updated ciphertext header $\langle q, \mathcal{C}_{DEK'} \rangle$. The cloud server updates all the involved data in its storage.

There are two functional requirements for our proposed scheme.

Correctness: For any integer λ and graph G with n vertices, any index $j \in \{1, \cdots, n\}$, any index k s.t. $SC_k \preceq SC_j$ in G and any message m,

$$Pr[\textbf{Decrypt}(G, M, param, j, dk_j, \langle k, \mathcal{C}_m \rangle) = m] = 1$$

where

$$(\{\gamma_i\}, param, M) \leftarrow \textbf{Setup}(1^\lambda, G),$$
$$dk_j \leftarrow \textbf{KeyGen}(G, param, j, \{\gamma_i\}),$$
$$ek_k \leftarrow \textbf{KeyGen}(G, param, k, \{\gamma_i\}),$$
$$\mathcal{C}_m \leftarrow \textbf{Encrypt}(param, m, k, ek_k).$$

Compactness: For any integer λ and graph G with n vertices, any index $j \in \{1, \cdots, n\}$ and any message m,

$$(\{\gamma_i\}, param, M) \leftarrow \textbf{Setup}(1^\lambda, G),$$
$$(dk_j, ek_j) \leftarrow \textbf{KeyGen}(G, param, j, \{\gamma_i\}),$$
$$\mathcal{C}_m \leftarrow \textbf{Encrypt}(param, m, k, ek_j),$$

$|dk_j|$ and $|\mathcal{C}_m|$ only depend on the security parameter λ but are independent of the number of vertices n in G.

4.2 Concrete Construction

The concrete construction of the six algorithms (**Setup**, **KeyGen**, **Encrypt**, **Decrypt**, **Update**, **ReEncrypt**) mentioned above are defined as follows.

Setup $(1^\lambda, G)$: $G = (V, E)$ is a DAG representing the hierarchical structure of user groups, where $V = \{SC_i \mid 1 \le i \le n\}$ is a set of vertices representing the n user groups in the system and E is a set of edges representing the partial order relations among user groups.

Randomly pick a bilinear group $\mathbb{G}$ of prime order

p where $2^{\lambda} \le p \le 2^{\lambda+1}$ and a generator $g \in \mathbb{G}$. Randomly pick a secret parameter $\alpha \in_R \mathbb{Z}_p$ as the system master secret key *msk* which should be kept secret by CA. Compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \cdots, n, n+2, \cdots, 2n$. Set the system parameter as

$$param = \langle g, g_1, \cdots, g_n, g_{n+2}, \cdots, g_{2n} \rangle .$$

Randomly pick n random numbers $\{\gamma_i \mid 1 \le i \le n\} \in_R \mathbb{Z}_p$ for n user groups in the system which should be kept secret by CA as well. For simplicity, let $\mathcal{S}_j$ denote the set $\{i \mid SC_i \preceq SC_j\}$ for each SC_j in G which means the set of all subordinate groups (including itself) of group j. Set a $n \times n$ public matrix

$$M = \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & t_{2,3} & \cdots & t_{2,n} \\ t_{3,1} & t_{3,2} & t_{3,3} & \cdots & t_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n,1} & t_{n,2} & t_{n,3} & \cdots & t_{n,n} \end{pmatrix}$$

where $t_{j,k} = \dfrac{\gamma_k}{h_j}$ ($h_j = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_i)$, $\mathcal{H}(\cdot): \mathbb{Z}_p \to \mathbb{Z}_p$ is a collision-resistant hash function mapping from $\mathbb{Z}_p$ to $\mathbb{Z}_p$) if $SC_k \preceq SC_j$; otherwise, $t_{j,k} = 0$.

KeyGen $(G, param, j, \{\gamma_i\})$: For the set of user groups which includes SC_j and all its descendant vertices in graph G, i.e., $\mathcal{S}_j = \{i \mid SC_i \preceq SC_j\}$, set $h_j = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_i)$ and compute

$$dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j}$$

as the private decryption key for each group j. Set the public encryption keys of each group j as $ek_j = g^{\gamma_j}$.

Encrypt $(param, m, k, ek_k)$: This algorithm mainly supports the function of scalable data sharing. For a message $m \in \mathbb{G}_T$ whose sharing target is group k, randomly pick $t \in_R \mathbb{Z}_p$ and compute the ciphertext as

$$\mathcal{C}_m = \langle g^t, (g^{\gamma_k} g_k)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle .$$

Decrypt $(G, M, param, j, dk_j, \langle k, \mathcal{C}_m \rangle)$: Only the users in the sharing target group k and its superior groups can decrypt the ciphertext $\mathcal{C}_m$.

If $SC_k \not\preceq SC_j$ in graph G, the users in group j cannot decrypt the ciphertext $\mathcal{C}_m$ with key dk_j and the algorithm returns $\perp$;

If $SC_k \preceq SC_j$ in graph G, the users in group j can decrypt the ciphertext $\mathcal{C}_m = \langle c_1, c_2, c_3 \rangle$ with key dk_j as

$$m_d = c_3 \cdot \frac{\hat{e}(dk_j^{t_{j,k}} \cdot \prod_{i \in S_j, i \ne k} g_{n+1-i+k}, c_1)}{\hat{e}(\prod_{i \in S_j} g_{n+1-i}, c_2)}$$

where $t_{j,k} = \dfrac{\gamma_k}{h_j}$ which can be obtained from the public matrix M and the algorithm returns m_d.

Update $(G, msk = \alpha, \{\gamma_i\}, param, l)$: Randomly pick new random numbers $\{\gamma_{q'} \mid SC_q \preceq SC_l\} \in_R \mathbb{Z}_p$ for group l which needs to be updated and all its descendant groups $\{q \mid SC_q \preceq SC_l\}$ in G, which should be kept secret by CA. Update the public encryption keys of group l and all its descendant groups as $\{ek_{q'} = g^{\gamma_{q'}}\}$. Update the element related to $\{\gamma_{q'}\}$ in the public matrix M and generate a new public matrix M'. Compute new private decryption keys for group l and all its descendant groups as well as their ascendant groups $\{k\} = \{q\} \cup \{j \mid SC_j \succeq SC_q\}$ as

$$\left\{ dk_{k'} = \prod_{i \in \mathcal{S}_k} g_{n+1-i}^{h_{k'}} \right\}$$

where $h_{k'} = \mathcal{H}(\sum_{i \in \mathcal{S}_k} \gamma_{i'})$, and generate re-encryption tokens for groups $\{q \mid SC_q \preceq SC_l\}$ as

$$\left\{ rk_q = \frac{\alpha^q + \gamma_{q'}}{\alpha^q + \gamma_q} \right\}.$$

ReEncrypt $(\langle q, C_m \rangle, rk_q)$: Update the $c_2 = (g^{\gamma_q} g_q)^t$ in all $C_m = \langle c_1, c_2, c_3 \rangle = \langle g^t, (g^{\gamma_q} g_q)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle$ of group q as $c_{2'} = c_2^{(\alpha^q + \gamma_{q'})/(\alpha^q + \gamma_q)} = ((g^{\gamma_q} g_q)^t)^{(\alpha^q + \gamma_{q'})/(\alpha^q + \gamma_q)} = (g^{\gamma_{q'}} g_q)^t$ with re-encryption token $rk_q = \dfrac{\alpha^q + \gamma_{q'}}{\alpha^q + \gamma_q}$ and obtain the updated ciphertext

$$\mathcal{C}_{m'} = \langle g^t, (g^{\gamma_{q'}} g_q)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle .$$

4.3 Dynamic Updates

Besides user revocation, here we give some tips for other dynamic updates of hierarchical user structure.

- *Relation Insertion*: If a new relation (SC_i, SC_j) such that $SC_j \prec SC_i$ needs to be added to the hierarchical

user structure G, for all groups $\{k \mid SC_k \succeq SC_i\}$, their decryption keys $\{dk_k\}$ and the related elements in the public matrix M should be updated to include their new subordinate groups $\{l \mid SC_l \preceq SC_j\}$ into their access privileges.

- *Relation Deletion*: If a relation (SC_i, SC_j) such that $SC_j \prec SC_i$ needs to be deleted from the hierarchical user structure G, for all groups $\{k \mid SC_k \succeq SC_i\}$, their decryption keys $\{dk_k\}$ and elements related to them in the public matrix M should be updated to exclude their former subordinate groups $\{l \mid SC_l \preceq SC_j\}$ from their access privileges. To ensure the forward security, **Update** and **ReEncrypt** algorithms for groups $\{l \mid SC_l \preceq SC_j\}$ should be invoked, in order to prevent groups $\{k \mid SC_k \succeq SC_i\}$ from accessing data of their former subordinate groups $\{l \mid SC_l \preceq SC_j\}$ with their old keys and public parameters.
- *Class Insertion*: The class insertion problem is considered to be solved by the extension scheme described in [41]. Or in a more general way, in the earlier system initialization step, n can be set as a larger number to reserve enough places for newly added classes afterward. When a new class needs to be added into G, CA first generates or updates all keys and system parameters related to it by allocating a reserved index i, and then inserts the incoming and outgoing relations of the new class SC_i by the relation insertion method above.
- *Class Deletion*: The deletion procedure of a class SC_i can be done by revoking all users in SC_i via **Update** and **ReEncrypt** algorithms, or deleting each of the incoming and outgoing relations of SC_i by the relation deletion method above.

5 Security Analysis

In this section, we analyze the five functional and security requirements mentioned previously namely *correctness*, *compactness*, *data confidentiality*, *collusion resistant* and *forward security* as follows.

Correctness: For the ciphertext of message m of group k

$$\mathcal{C}_m = \langle g^t, (g^{\gamma_k} g_k)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle,$$

we can see that

$$m_d = c_3 \cdot \frac{\hat{e}(dk_j^{t_{j,k}} \cdot \prod_{i \in \mathcal{S}_j, i \neq k} g_{n+1-i+k}, c_1)}{\hat{e}(\prod_{i \in \mathcal{S}_j} g_{n+1-i}, c_2)}$$

$$= c_3 \cdot \frac{\hat{e}(dk_j^{\gamma_k / h_j} \cdot \prod_{i \in \mathcal{S}_j, i \neq k} g_{n+1-i+k}, g^t)}{\hat{e}(\prod_{i \in \mathcal{S}_j} g_{n+1-i}, (g^{\gamma_k} g_k)^t)}$$

$$= c_3 \cdot \frac{\hat{e}(\prod_{i \in \mathcal{S}_j, i \neq k} g_{n+1-i+k}, g^t)}{\hat{e}(\prod_{i \in \mathcal{S}_j} g_{n+1-i}, g_k^t)}$$

$$= c_3 \cdot \frac{\hat{e}(\prod_{i \in \mathcal{S}_j} g_{n+1-i+k}, g^t) / \hat{e}(g_{n+1}, g^t)}{\hat{e}(\prod_{i \in \mathcal{S}_j} g_{n+1-i}, g_k^t)}$$

$$= m \cdot \frac{\hat{e}(g_1, g_n)^t}{\hat{e}(g_{n+1}, g^t)} = m.$$

Therefore, our concrete scheme is said to be correct since for any message m of group k which satisfies

$$(\{\gamma_i\}, param, M) \leftarrow \textbf{Setup}(1^\lambda, G),$$
$$dk_j \leftarrow \textbf{KeyGen}(G, param, j, \{\gamma_i\}),$$
$$ek_k \leftarrow \textbf{KeyGen}(G, param, k, \{\gamma_i\}),$$
$$\mathcal{C}_m \leftarrow \textbf{Encrypt}(param, m, k, ek_k),$$

there is $Pr[\textbf{Decrypt}(G, M, param, j, dk_j, \langle k, \mathcal{C}_m \rangle) = m] = 1$.

Compactness: All calculations in our concrete scheme are in the finite field $\mathbb{Z}_p$ or group $\mathbb{G}$, $\mathbb{G}_T$ of prime order p, where $2^\lambda \leq p \leq 2^{\lambda+1}$.

For $dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j}$, since $h_j = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_i) \in \mathbb{Z}_p$ ($\{\gamma_i\} \in \mathbb{Z}_p$) and $g_{n+1-i} = g^{\alpha^{n+1-i}} \in \mathbb{G}$ ($\alpha \in \mathbb{Z}_p$), $\prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j} \in \mathbb{G}$,

$$\left| \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j} \right| \leq \lambda + 1.$$

For $\mathcal{C}_m = \langle g^t, (g^{\gamma_k} g_k)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle$, since $g^t \in \mathbb{G}$ ($t \in \mathbb{Z}_p$),

$$\left| g^t \right| \leq \lambda + 1;$$

since $g^{\gamma_k} \in \mathbb{G}$ ($\gamma_k \in \mathbb{Z}_p$) and $g_k = g^{\alpha^k} \in \mathbb{G}$ ($\alpha \in \mathbb{Z}_p$), $(g^{\gamma_k} g_k)^t \in \mathbb{G}$ ($t \in \mathbb{Z}_p$),

$$\left| (g^{\gamma_k} g_k)^t \right| \leq \lambda + 1;$$

since $\hat{e}(g_1, g_n) \in \mathbb{G}_T$, $\hat{e}(g_1, g_n)^t \in \mathbb{G}_T$ ($t \in \mathbb{Z}_p$),

$$\left|\hat{e}(g_1, g_n)^t\right| \le \lambda + 1.$$

Therefore, the sizes of sk_i and $\mathcal{C}_m$ only depend on the security parameter λ input in **Setup** rather than the number of vertices n in G.

Data Confidentiality: The data confidentiality of our proposed scheme is based on the KAC scheme [41] which is proved to be semantically secure under the bilinear Diffie-Hellman Exponent assumption (BDHE) and collusion-resistant in [42].

In our scheme, if a user in group j wants to access the data of group k ($SC_k \not\preceq SC_j$) with his/her decryption key $dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j}$, he/she needs to know $g_{n+1-k}^{h_j}$ to compute a new secret key

$$dk_{j'} = g_{n+1-k}^{h_j} \cdot dk_j = g_{n+1-k}^{h_j} \cdot \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j} = \prod_{i \in \mathcal{S}_j \cup \{k\}} g_{n+1-i}^{h_j}$$

which includes the decryption privilege of group k 's data. However, since $h_j = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_i)$ and the system parameters $\{\gamma_i\}$ are kept secret by CA, the user cannot calculate $g_{n+1-k}^{h_j}$ by himself/herself.

In addition, $t_{j,k} = \frac{\gamma_k}{h_j}$ is also needed for **Decrypt** procedure while actually $t_{j,k} = 0$ in the public matrix M because $SC_k \not\preceq SC_j$.

Therefore, our concrete scheme can ensure the data confidentiality since any user in group j cannot access the data of group k if $SC_k \not\preceq SC_j$.

Collusion Resistant: Suppose SC_j is a user group at a higher level and $SC_{j_1}, SC_{j_2}, \cdots, SC_{j_m}$ are SC_j 's descendant groups. If users in $SC_{j_1}, SC_{j_2}, \cdots, SC_{j_m}$ want to collude to decrypt the data of SC_j, they need to derive

$$dk_j^{t_{j,j}} = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{\gamma_j}$$

by their secret keys $dk_{j_1}, dk_{j_2}, \cdots, dk_{j_m}$ and the public matrix M.

However, since $SC_j \not\preceq SC_{j_1}, SC_{j_2}, \cdots, SC_{j_m}$, these colluded users cannot obtain each $g_{n+1-i}^{\gamma_j}$ by computing $(g_{n+1-i}^{h_{j_k}})^{t_{j_k,j}} = (g_{n+1-i}^{h_{j_k}})^{\gamma_j / h_{j_k}}$ because $t_{j_k,j} = 0$ in M, although $g_{n+1-i}^{h_{j_k}}$ may be obtained by users in SC_{j_k} from $dk_{j_k} = \prod_{i \in \mathcal{S}_{j_k}} g_{n+1-i}^{h_{j_k}} = g_{n+1-i}^{h_{j_k}}$ if SC_{j_k} has no descendant group.

Therefore, our concrete scheme is said to be collusion-resistant.

Forward Security: For a user u in group j with secret key $dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j}$, once his/her access privilege is revoked from SC_j, the system parameters and secret keys related to group j and all its descendant groups $\{k \mid SC_k \preceq SC_j\}$ will be updated as

$$\{\gamma_k\} \to \{\gamma_{k'}\},$$

$$h_j = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_i) \to h_{j'} = \mathcal{H}(\sum_{i \in \mathcal{S}_j} \gamma_{i'}),$$

$$\left\{t_{j,k} = \frac{\gamma_k}{h_j}\right\} \to \left\{t_{j,k'} = \frac{\gamma_{k'}}{h_{j'}}\right\},$$

$$dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j} \to dk_{j'} = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_{j'}},$$

and the data of these groups will be re-encrypted into

$$\left\{\mathcal{C}_{m'} = \langle g^t, (g^{\gamma_{k'}} g_q)^t, m \cdot \hat{e}(g_1, g_n)^t \rangle\right\}.$$

Then, the user u cannot use his/her old secret key $dk_j = \prod_{i \in \mathcal{S}_j} g_{n+1-i}^{h_j}$ to decrypt the updated data $\{\mathcal{C}_{m'}\}$ by **Decrypt** unless he/she knows $\left\{\frac{\gamma_{k'}}{h_j}\right\}$ which would never appear in the public matrix M. Therefore, our scheme can ensure the forward security.

6 Performance Analysis

In this section, we first compare the features of our scheme with others. Then, we discuss the performance of our scheme by analyzing and simulation.

6.1 Comparison

We make a comparison of our proposed scheme with some other hierarchical access control schemes of Akl and Taylor [19], Sandhu [20], Damiani et al. [21], Atallah et al. [22] and Tang et al. [18] in Table 2.

Table 2. Comparison with related schemes

Schemes	Cryptosystem	Supported hierarchy	Key number of each user	Key size of single key	Data sharing	Key derivation	Delegated re-encryption
Akl-Taylor [19]	Symmetric	DAG	Single	Non-constant	No	Yes	No
Sandhu [20]	Symmetric	Tree	Single	Constant	No	Yes	No
Damiani *et al*. [21]	Symmetric	Tree	Multiple	Constant	No	Yes	No
Atallah *et al*. [22]	Symmetric	DAG	Single	Constant	No	Yes	No
Tang *et al*. [18]	Symmetric	DAG	Single	Constant	No	Yes	No
Our Scheme	Public-key	DAG	Single	Constant	Yes	No	Yes

All of these other schemes are based on symmetric cryptosystem while our scheme is a public-key one. Both schemes of Sandhu [20] and Damiani et al. [21] only support tree as the hierarchical structure while others support DAG which beyond the scope of the tree structure. In the scheme of Damiani et al. [21], each user should keep several keys which may cause the inconvenient of key management. In Akl-Taylor scheme [19], the size of each key is non-constant which is not suitable for a large-scale hierarchy.

The most significant advantage of our scheme is that we support scalable data sharing and delegated re-encryption which is more suitable for the cloud environment and can reduce the computational overhead of cloud users. Moreover, the key derivation procedure is not needed in our scheme which means that users can decrypt the data of their subordinate groups directly without deriving their descendant keys.

6.2 Computation Complexity

Since the schemes listed above are under the symmetric cryptosystem, it is meaningless to compare them with our scheme in computation complexity. Therefore, we compare our scheme with the hierarchical access control scheme of Chen et al. [25], which is also based on bilinear pairings, in terms of the computation complexity on a user when performing encryption and decryption. Suppose n is the number of user groups in hierarchical user structure, j is the index of the group where the current user is and $|\mathcal{S}_j|$ is the number of group j's descendant groups (including itself) ($\mathcal{S}_j = \{i \mid SC_i \preceq SC_j\}$). The time cost of multiplication operation in $\mathbb{G}$, multiplication operation in $\mathbb{G}_T$, multiplication operation in $\mathbb{Z}_p$, exponentiation operation in $\mathbb{G}$, exponentiation operation in $\mathbb{G}_T$, inverse operation in $\mathbb{G}_T$, inverse operation in $\mathbb{Z}_p$, pairing operation and the cryptographic hash function are denoted by T_{m_1}, T_{m_2}, T_{m_3}, T_{e_1}, T_{e_2}, T_{i_1}, T_{i_2}, T_p and T_h respectively. Besides, other operations like addition operation in $\mathbb{Z}_p$ are omitted here.

In the data encryption phase, the time cost of each encryption operation for a user in group j in our scheme is $T_{m_1} + T_{m_2} + 2T_{e_1} + T_{e_2} + T_p$ which is $\mathcal{O}(1)$. In addition, since $\hat{e}(g_1, g_n)$ in **Encrypt** algorithm can be pre-computed in the process of system initialization as a public parameter in our scheme, the time cost can be reduced into $T_{m_1} + T_{m_2} + 2T_{e_1} + T_{e_2}$. In the scheme of Chen *et al*. [25], the time cost of each encryption operation for a user in group j is $T_{e_1} + 2T_{e_2} + T_{m_2} + (|\mathcal{S}_j|)T_{m_3} + (|\mathcal{S}_j|)T_h$ which is $\mathcal{O}(|\mathcal{S}_j|)$. That is to say, our scheme is more efficient in the data encryption phase.

In the data decryption phase, the time cost of each decryption operation for a user in group j in our scheme is $(2 \cdot |\mathcal{S}_j| - 2)T_{m_1} + 2T_{m_2} + T_{e_1} + T_{i_1} + 2T_p$ which is $\mathcal{O}(|\mathcal{S}_j|)$. In the scheme of Chen *et al*. [25], the time cost of each decryption operation for a user in group j is $2T_{m_2} + T_{e_2} + T_{i_1} + T_{i_2} + 2T_p$ which is $\mathcal{O}(1)$. In the real-world application scenarios, however, most users are in bottom groups with no subordinate user. For these users, $|\mathcal{S}_j|$ is 1 so that the efficiency of our scheme is also acceptable in the data decryption phase.

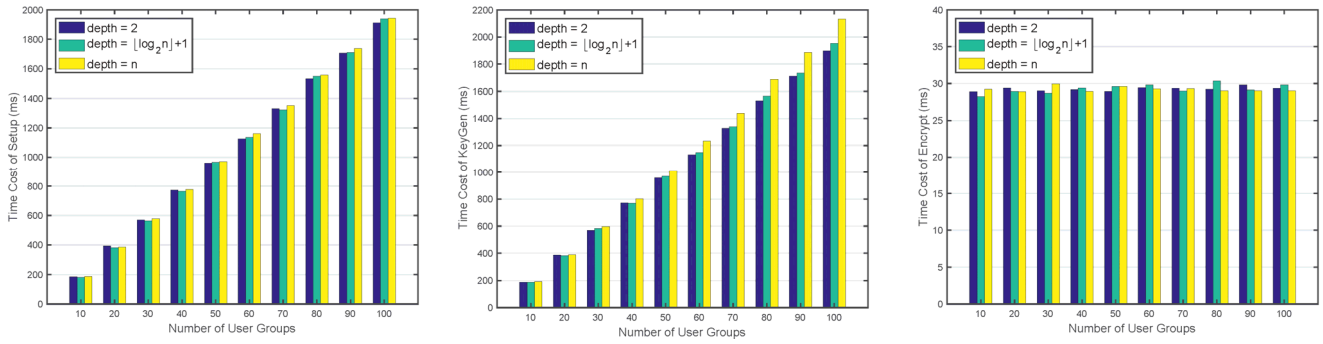
6.3 Implementation and Experimental Evaluation

For simplicity, we can use tree structure instead of DAG to represent the hierarchy of user groups since it would not influent the performance of our scheme but can be more comprehensible in programming. Actually, the public matrix M can take place of the hierarchical user structure G in all procedures after its generation, since for any two groups i and j, $SC_i \preceq SC_j$ can be determined by checking whether $t_{j,i} \neq 0$ in M.

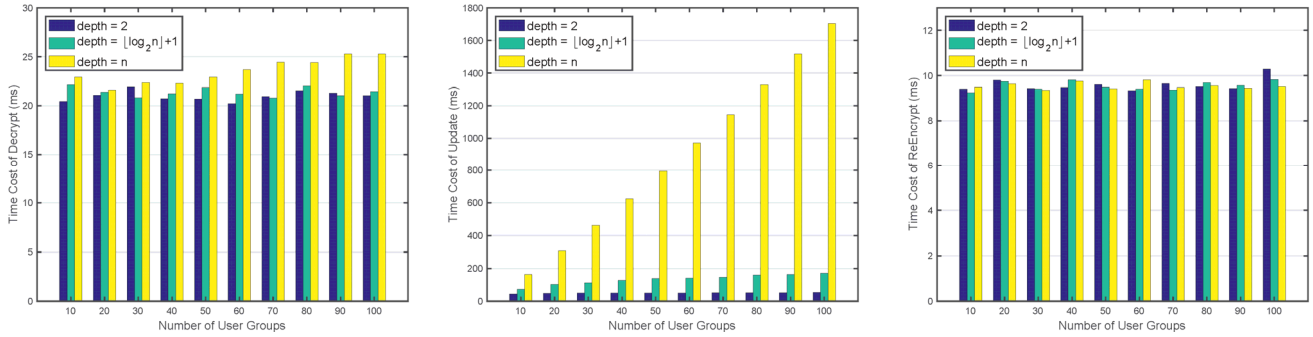
We implemented the proposed scheme to evaluate its performance using Java programming language with the Java Pairing-Based Cryptography Library (JPBC) [43] on an Ubuntu 14.04 LTS desktop computer with Intel Xeon CPU E5-1620 v3 at 3.5 GHz and 16 GB RAM. We utilized the SHA-256 algorithm as the hash function $\mathcal{H}(\cdot)$.

We tested the time cost of **Setup**, **KeyGen**, **Encrypt**, **Decrypt**, **Update** and **ReEncrypt** of our

scheme shown in Figure 4. In our implementation, we fixed the message size as 256 bits in testing the time cost of each phase of our scheme since the message of the public-key algorithm is the secret key of the symmetric algorithm (e.g. AES) in hybrid encryption. In our evaluation, the total number of user groups noted as n is vary from 10 to 100. We compared the average time cost of each phase in three typical conditions of hierarchical structure including the full $(n-1)$ -ary tree with depth of 2, the complete binary tree with depth of $\lfloor log_2 n \rfloor + 1$ and the chain (totally ordered set) with depth of n.



(a) Time cost of **Setup** (b) Time cost of **KeyGen** (c) Time cost of **Encrypt**



(d) Time cost of **Decrypt** (e) Time cost of **Update** (f) Time cost of **ReEncrypt**

Figure 4. Time cost of each algorithm in our concrete scheme

In Figure 4(a) and Figure 4(b), we can see that the time cost of **Setup** and **KeyGen** increase with the increasing of user groups number n. However, it is acceptable since **Setup** and **KeyGen** will be executed only once at the beginning. In Figure 4(c), Figure 4(d), Figure 4(f), the average time cost of **Encrypt**, **Decrypt** and **ReEncrypt** are relatively stable with the increasing of n. In Figure 4(e), the average time cost of **Update** when user revocation occurs increases rapidly if the depth of the hierarchical structure is n, while it increases much more slowly if the depth of the hierarchical structure is lower. Fortunately, the average time cost of **Update** is much lower than that of the extreme case because the hierarchical user structure in normal scenarios is often a large-scale tree with limited depth.

7 Conclusion

In this paper, we propose a hierarchical access control scheme in cloud storage under the public-key cryptosystem, which supports scalable data sharing and delegated data re-encryption. Our scheme is mainly based on KAC, in which a user is assigned with an aggregate key including the decryption privileges of the group where the user is and all its subordinate groups in the hierarchical user structure. To achieve the scalable data sharing in our hierarchical access control framework, any user can encrypt data into the ciphertext of any group and upload the encrypted data to cloud storage. In addition, our scheme delegates the re-encryption tasks to the cloud server in order to reduce the computation and communication overhead of users, which allows the cloud server to update the encrypted data in its storage when user revocation occurs. The analyses of security and performance indicate that our scheme is functional, secure and

efficient.

In the future, we will focus on how to achieve traceable anonymous data sharing and collaboration in hierarchical access control utilizing techniques such as group signature, ring signature or chameleon hashing and signatures [44].

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 61572382 and No. 61702401), Key Project of Natural Science Basic Research Plan in Shaanxi Province of China (No. 2016JZ021), China 111 Project (No. B16037), Guangxi Cooperative Innovation Center of Cloud Computing and Big Data (No. YD17X07), Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems (No. YF17103), and the Fundamental Research Funds for the Central Universities (No. XJS17053 and No. JBF181501).

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility, *Future Generation Computer Systems*, Vol. 25, No. 6, pp. 599-616, June, 2009.

[2] X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New Algorithms for Secure Outsourcing of Modular Exponentiations, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 9, pp. 2386-2396, September, 2014.

[3] X. Zhang, T. Jiang, K.-C. Li, A. Castiglione, X. Chen, New Publicly Verifiable Computation for Batch Matrix Multiplication, *Information Sciences*, December, 2017.

[4] J. Wang, X. Chen, X. Huang, I. You, Y. Xiang, Verifiable Auditing for Outsourced Database in Cloud Computing, *IEEE Transactions on Computers*, Vol. 64, No. 11, pp. 3293-3303, November, 2015.

[5] T. Jiang, X. Chen, J. Ma, Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation, *IEEE Transactions on Computers*, Vol. 65, No. 8, pp. 2363-2373, August, 2016.

[6] J. Shen, J. Shen, X. Chen, X. Huang, W. Susilo, An Efficient Public Auditing Protocol with Novel Dynamic Structure for Cloud Data, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 10, pp. 2402-2415, October, 2017.

[7] Y. Ren, J. Shen, J. Wang, J. Han, S. Lee, Mutual Verifiable Provable Data Auditing in Public Cloud Storage, *Journal of Internet Technology*, Vol. 16, No. 2, pp. 317-323, March, 2015.

[8] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, W. Lou, Secure and Efficient Cloud Data Deduplication with Randomized Tag, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 3, pp. 532-543, March, 2017.

[9] H. Yuan, X. Chen, T. Jiang, X. Zhang, Z. Yan, Y. Xiang, DedupDUM: Secure and Scalable Data Deduplication with Dynamic User Management, *Information Sciences*, Vol. 456, pp. 159-173, August, 2018.

[10] J. Wang, M. Miao, Y. Gao, X. Chen, Enabling Efficient Approximate Nearest Neighbor Search for Outsourced Database in Cloud Computing, *Soft Computing*, Vol. 20, No. 11, pp. 4487-4495, November, 2016.

[11] J. Wang, X. Chen, J. Li, J. Zhao, J. Shen, Towards Achieving Flexible and Verifiable Search for Outsourced Database in Cloud Computing, *Future Generation Computer Systems*, Vol. 67, pp. 266-275, February, 2017.

[12] A. Sahai, B. Waters, Fuzzy Identity-based Encryption, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Aarhus, Denmark, 2005, pp. 457-473.

[13] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based Encryption for Fine-grained Access Control of Encrypted Data, *ACM Conference on Computer and Communications Security*, Alexandria, VA, 2006, pp. 89-98.

[14] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-Policy Attribute-Based Encryption, *IEEE Symposium on Security and Privacy*, Oakland, CA, 2007, pp. 321-334.

[15] S. Yu, C. Wang, K. Ren, W. Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, *IEEE International Conference on Computer Communications*, San Diego, CA, 2010, pp. 534-542,.

[16] J. Li, X. Huang, J. Li, X. Chen, Y. Xiang, Securely Outsourcing Attribute-Based Encryption with Checkability, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 8, pp. 2201-2210, August, 2014.

[17] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, I. You, Ensuring Attribute Privacy Protection and Fast Decryption for Outsourced Data Security in Mobile Cloud Computing, *Information Sciences*, Vol. 379, pp. 42-61, February, 2017.

[18] S. Tang, X. Li, X. Huang, Y. Xiang, L. Xu, Achieving Simple, Secure and Efficient Hierarchical Access Control in Cloud Computing, *IEEE Transactions on Computers*, Vol. 65, No. 7, pp. 2325-2331, July, 2016.

[19] S. G. Akl, P. D. Taylor, Cryptographic Solution to a Problem of Access Control in a Hierarchy, *ACM Transactions on Computer Systems*, Vol. 1, No. 3, pp. 239-248, August, 1983.

[20] R. S. Sandhu, Cryptographic Implementation of a Tree Hierarchy for Access Control, *Information Processing Letters*, Vol. 27, No. 2, pp. 95-98, February, 1988.

[21] E. Damiani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, Key Management for Multi-User Encrypted Databases, *ACM Workshop on Storage Security and Survivability*, Fairfax, VA, 2005, pp. 74-83.

[22] M. J. Atallah, M. Blanton, N. Fazio, K. B. Frikken, Dynamic and Efficient Key Management for Access Hierarchies, *ACM Transactions on Information System Security*, Vol. 12, No. 3, pp. 18:1-18:43, January, 2009.

[23] J. Crampton, K. M. Martin, P. R. Wild, On Key Assignment for Hierarchical Access Control, *IEEE Computer Security Foundations Workshop*, 2006, Venice, Italy, pp. 98-111.

[24] Y.-R. Chen, C.-K. Chu, W.-G. Tzeng, J. Zhou, CloudHKA: A

Cryptographic Approach for Hierarchical Access Control in Cloud Computing, *International Conference on Applied Cryptography and Network Security*, Banff, Canada, 2013, pp. 37-52.

[25] Y.-R. Chen, W.-G. Tzeng, Hierarchical Key Assignment with Dynamic Read-Write Privilege Enforcement and Extended KI-Security, *International Conference on Applied Cryptography and Network Security*, Kanazawa, Japan, 2017, pp. 165-183.

[26] A. Castiglione, A. D. Santis, B. Masucci, F. Palmieri, A. Castiglione, J. Li, X. Huang, Hierarchical and Shared Access Control, *IEEE Transactions on Information Forensics and Security*, Vol. 11, No. 4, pp. 850-865, April, 2016.

[27] A. Castiglione, A. D. Santis, B. Masucci, F. Palmieri, A. Castiglione, X. Huang, Cryptographic Hierarchical Access Control for Dynamic Structures, *IEEE Transactions on Information Forensics and Security*, Vol. 11, No. 10, pp. 2349-2364, October, 2016.

[28] A. Castiglione, A. D. Santis, B. Masucci, F. Palmieri, X. Huang, A. Castiglione, Supporting Dynamic Updates in Storage Clouds with the Akl-Taylor Scheme, *Information Sciences*, Vol. 387, pp. 56-74, May, 2017.

[29] J. Alderman, N. Farley, J. Crampton, Tree-based Cryptographic Access Control, *European Symposium on Research in Computer Security*, Oslo, Norway, 2017, pp. 47-64.

[30] Z. Zhang, X. Chen, J. Li, X. Tao, J. Ma, HVDB: A Hierarchical Verifiable Database Scheme with Scalable Updates, *Journal of Ambient Intelligence and Humanized Computing*, March, 2018.

[31] X. Chen, J. Li, X. Huang, J. Ma, W. Lou, New Publicly Verifiable Databases with Efficient Updates, *IEEE Transactions on Dependable and Secure Computing*, Vol. 12, No. 5, pp. 546-556, September, 2015.

[32] X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable Computation over Large Database with Incremental Updates, *IEEE Transactions on Computers*, Vol. 65, No. 10, pp. 3184-3195, October, 2016.

[33] D. Catalano, D. Fiore, Vector Commitments and their Applications, *International Conference on Practice and Theory in Public-Key Cryptography*, Nara, Japan, 2013, pp. 55-72.

[34] G. Wang, Q. Liu, J. Wu, Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services, *ACM Conference on Computer and Communications Security*, Chicago, IL, 2010, pp. 735-737.

[35] Z. Wan, J. Liu, R. H. Deng, HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing, *IEEE Transactions on Information Forensics and Security*, Vol. 7, No. 2, pp. 743-754, April, 2012.

[36] Q. Huang, Y. Yang, M. Shen, Secure and Efficient Data Collaboration with Hierarchical Attribute-Based Encryption in Cloud Computing, *Future Generation Computer Systems*, Vol. 72, pp. 239-249, July, 2017.

[37] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, Y. Xiang, Block Design-Based Key Agreement for Group Data Sharing in Cloud Computing, *IEEE Transactions on Dependable and Secure Computing*, July, 2017.

[38] J. Shen, T. Zhou, X. Chen, J. Li, W. Susilo, Anonymous and Traceable Group Data Sharing in Cloud Computing, *IEEE Transactions on Information Forensics and Security*, Vol. 13, No. 4, pp. 912-925, April, 2018.

[39] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure Attribute-Based Data Sharing for Resource-Limited Users in Cloud Computing, *Computers & Security*, Vol. 72, pp. 1-12, January, 2018.

[40] Z. Zhang, X. Chen, J. Ma, J. Shen, SLDS: Secure and Location-Sensitive Data Sharing Scheme for Cloud-Assisted Cyber-Physical Systems, *Future Generation Computer Systems*, February, 2018.

[41] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, R. H. Deng, Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 2, pp. 468-477, February, 2014.

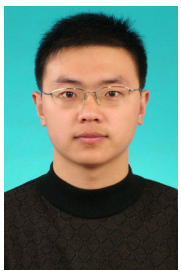
[42] D. Boneh, C. Gentry, B. Waters, Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys, *Annual International Cryptology Conference*, Santa Barbara, CA, 2005, pp. 258-275.

[43] A. D. Caro, V. Iovino, jPBC: Java Pairing Based Cryptography, *IEEE Symposium on Computers and Communications*, Kerkyra, Greece, 2011, pp. 850-855.

[44] X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, K. Kim, Identity-Based Chameleon Hashing and Signatures without Key Exposure, *Information Sciences*, Vol. 265, pp. 198-210, May, 2014.

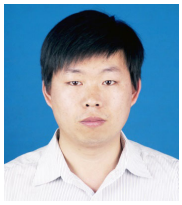
Biographies

Zhenyao Qiu received his B.S. degree from Central China Normal University in 2016. He is currently working toward his M.S. degree in information security at Xidian University, China. His research interests include data security and access control in cloud computing.

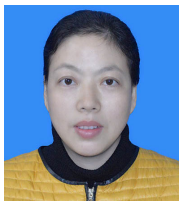


Zhiwei Zhang received his B.S. degree in network engineering and M.S. degree in computer systems architecture from Xidian University, China, in 2008 and 2011, respectively. His research interests include data secure management, data storage security and data location verification in cloud computing.

Shichong Tan received his B.S. degree in telecommunications engineering, M.S. and Ph.D. degrees in cryptography from Xidian University, China, in 2002, 2005 and 2009, respectively. Since 2005, he has been with Xidian University, where he is now an associate professor. His research interests include cloud computing, bitcoin and blockchain technologies.



Jianfeng Wang received his M.S. degree in mathematics and Ph.D. degree in cryptography from Xidian University, China, in 2013 and 2016, respectively. Since 2016, he has been with Xidian University. His research interests include applied cryptography and cloud security.



Xiaoling Tao received her B.S. and M.S. degrees from Guilin University of Electronic Technology, China, in 2000 and 2008, respectively. Since 2000, she has been with Guilin University of Electronic Technology, where she is now a professor. Her research interests include cloud computing, big data and cyber security.