COMPUTERS & SECURITY 42 (2014) 151–164

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing

Xin Dong [a], Jiadi Yu [a,*], Yuan Luo [a], Yingying Chen [b], Guangtao Xue [a], Minglu Li [a]

[a] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, PR China
[b] Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA

ARTICLE INFO

Article history:
Received 7 May 2013
Received in revised form
30 October 2013
Accepted 18 December 2013

Keywords:
Data security
Data sharing
Privacy-preserving
Attribute-based encryption
Scalability
Cloud computing

ABSTRACT

Data sharing in the cloud, fueled by favorable trends in cloud technology, is emerging as a promising technique for allowing users to conveniently access data. However, the growing number of enterprises and customers who stores their data in cloud servers is increasingly challenging users' privacy and the security of data. This paper focuses on providing a dependable and secure cloud data sharing service that allows users dynamic access to their data. In order to achieve this, we propose an effective, scalable and flexible privacy-preserving data policy with semantic security, by utilizing ciphertext policy attribute-based encryption (CP-ABE) combined with identity-based encryption (IBE) techniques. In addition to ensuring robust data sharing security, our policy succeeds in preserving the privacy of cloud users and supports efficient and secure dynamic operations including, but not limited to, file creation, user revocation and modification of user attributes. Security analysis indicates that the proposed policy is secure under the generic bilinear group model in the random oracle model and enforces fine-grained access control, full collusion resistance and backward secrecy. Furthermore, performance analysis and experimental results show that the overheads are as light as possible.

1. Introduction

Cloud computing (Armbrust et al., 2009) is currently emerging as a technology in which cloud service providers (CSP) offer efficient data storage and computing facilities to a global client base. The only requirement for a user is a connected terminal. By employing a combination of virtualization techniques, service-oriented computing and other emerging technologies, cloud computing can be categorized into three types of "X as a service (XaaS)" pay-as-you-go services: the Platform as a Service (PaaS) model, e.g. Microsoft Azure (Mic), where users can deploy their own applications and tools to the cloud; Infrastructure as a Service (IaaS), e.g. Amazon EC2 (Ama), where users can utilize cloud services provided by the CSP to deploy arbitrary software; and Software as a Service (SaaS), e.g. Google App Engine (Goo), where users use applications provided by the CSP that run on the cloud infrastructure.

Storing data in the cloud offers users the convenience of access without requiring direct knowledge of the deployment and management of the hardware or infrastructure. Although cloud computing is much more powerful than personal

* Corresponding author. Tel./fax: +86 21 3420 5856.
E-mail address: jiadiyu@sjtu.edu.cn (J. Yu).
0167-4048/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2013.12.002

computing, it brings new privacy and security challenges, as users relinquish control by outsourcing their data they no longer having physical possession of it. By having full access to cloud services, users' data are exposed to a variety of threats and malicious attacks and cases of security breaches occur frequently (Arrington, 2006). For example, some clouds may be unfaithful to data confidentiality for monetary reasons; confidential information may be disclosed to business competitors; or the CSP may conceal data loss to maintain their reputation (Shah et al., 2007). In summary, although cloud computing is economically attractive to consumers and enterprises by offering users large-scale data sharing, it does not guarantee users privacy and data security.

Data owners demand high levels of security and confidentiality when they outsource their data to a cloud; although they usually encrypt their data when storing it in a cloud server, they still want control over it, for example, if they frequently update it (Erway et al., 2009; Ateniese et al., 2008). Direct employment of traditional cryptographic primitives cannot achieve the data security required. Thus, a considerable amount of work has recently been directed towards ensuring the privacy and security of remotely stored shared data using a variety of systems and security models (Yu et al., 2010a; Wang et al., 2010). These have mainly focused on preserving users' privacy while realizing desired security goals, without introducing excessively high levels of complexity to the users at the decryption stage. To solve these issues, researchers have either utilized key-policy attribute-based encryption (KP-ABE) for secure access control or employed hierarchical identity-based encryption (HIBE) for data security. Yu et al. (2010a) were the first team to achieve secure data access control with provable security in cloud computing using KP-ABE. However, by revealing some of the users' attributes to cloud, these systems were unable to fully preserve users' privacy. Conversely, the HIBE-based scheme (Wang et al., 2010) utilizes hierarchical encryption to ensure data security in a cloud, but this introduces too many private keys for each user to be managed efficiently. In summary, these schemes either have privacy flaws or provide security at the expense of performance; therefore, the challenge of achieving the dual goals of privacy-preserving with effective cloud data sharing remains unresolved.

To realize an effective, scalable and privacy-preserving data sharing service in cloud computing, the following challenges need to be met: firstly, data owners should be able to assign other cloud users with different access privileges to their data; secondly, the cloud needs to be able to support dynamic requests so that data owners can add or revoke access privileges to other users allowing them to create or delete their data; thirdly, the users' privacy must be protected against the cloud so that they can conceal their private information while accessing the cloud; finally, users should be able to access shared data in the cloud through connected technologies with low computing ability, such as smartphones and tablets. To date, solving these important areas in cloud computing remains elusive.

In this paper, we propose an effective, scalable and flexible privacy-preserving data sharing scheme in the cloud, that ensures both semantic security and effective availability of user data. To preserve privacy and guarantee data confidentiality against the cloud, the scheme employs a cryptographic primitive, named cipher-text policy attribute-based encryption (CP-ABE) and combines it with an identity-based encryption (IBE) technique; each data file is described by a set of meaningful attributes, allowing each user to be assigned an access structure that defines the scope of data files they can have access to. To enforce these access structures, this scheme defines a public-private key pair for each attribute. For each user' secret key, it is a combination of user's ID (i.e., user's public key) and the attribute's secret key, thereby ensuring that each attribute presents a different key to each user. Data files are encrypted by public key components and access matrices converted from the access structure; user secret keys are defined to reflect their access privileges so that a user can only decrypt a ciphertext if they have the matched attributes to satisfy the ciphertext. To resolve the challenging issues of collusion resistance, our scheme provides users with a public key fitted to their secret keys; we use user's ID (public key) to "tie" together the attributes belonging to this user so that they cannot be successfully combined with another's user's attributes. To protect user privacy, our scheme does not need to update user secret key so that it prevents cloud access user access structure. To reduce the key management issue, the data owner simply assigns secret keys to users via the cloud.

Compared to previous schemes, our proposed scheme provides the benefits of security and efficiency: 1) the cloud can learn nothing about a user's privacy or access structure, as such the scheme is fully collusion resistant; 2) all extended operations, including user revocation, can only affect the current file or user without involving key updates. Therefore, the main contributions of this paper can be summarized as follows:

1. Our scheme proposes effective, scalable encryption for a cloud data sharing service that simultaneously achieves full privacy-preserving, collusion resistance and data confidentiality.
2. We prove that the proposed scheme provides semantic security for data sharing in cloud computing through the random oracle under the generic bilinear group model (Boneh et al., 2005). Furthermore, our scheme simultaneously enforces fine-grandness, backward secrecy and access privilege confidentiality.
3. The performance analysis indicates that our scheme only incurs a small overhead compared to existing schemes; meanwhile, the experimental results demonstrate that the overheads are as light as possible.

The remainder of this paper is organized as follows: Section 2 discusses related works; Section 3 introduces the system model, adversary model, security requirements and our design goal; Section 4 provides the details of our scheme; Section 5 shows how our scheme can support file creation/deletion, user addition/revocation and modification of user attributes; Sections 6 and 7 analyze the security and performance of our scheme, respectively; finally, Section 8 provides the concluding remarks of the paper.

2. Related work

The concept of identity-based encryption (IBE) was proposed by Shamir (1985); however, a full IBE scheme was not

developed until 2001 (Boneh and Franklin, 2001; Cocks, 2001). IBE is a public-key cryptosystem (PKC) in which the public key assigned to each unique user is an arbitrary string similar to a user ID or email address; and a trusted third party, called the private key generator (PKG), calculates the corresponding private key. Compared to a traditional PKC, the IBE scheme eliminates the issue of searching for a recipient's public key, but has a key escrow problem.

An attribute-based encryption (ABE) system, first proposed by Sahai and Waters (2005). It is essentially a simplified IBE system with only a single attribute. In an ABE scheme, the sender encrypts the message with a set of attributes and specifies a number d; a recipient can only decrypt the encrypted message if they have at least d of the given attributes. Based on these principles, Goyal et al. (2006) proposed an ABE scheme with fine-grained data access control that supports monotonic access structures, such as AND, OR and other threshold gates. Ostrovsky et al. (2007) proposed an enhanced scheme that also supports non-monotonic access structures, i.e., NOT gates. There are two classes of ABE: key-policy attribute-based encryption (KP-ABE)((Goyal et al., 2006)); and ciphertext policy attribute-based encryption (CP-ABE), first introduced by Bethencourt et al. (2007). In KP-ABE, the access structure is used to encrypt the secret key, and the attributes are used to describe the ciphertext. Conversely, CP-ABE uses the access structure to encrypt the ciphertext and the secret key is generated based on an attribute set.

The ABE scheme proposed by Sahai and Water is the foundation of our scheme. Briefly, it consists of the following four algorithms:

1. System Initialization: the sender chooses the algebraic groups and several secure parameters.
2. Encryption: the sender encrypts the message using the access structure.
3. Key Generation: the sender generates secret keys for the recipients, depending on the set of attributes the recipients possess.
4. Decryption: the recipient decrypts the message with a valid set of attributes.

Müller et al. (2009) presented a distributed attribute-based scheme, based on an efficient construction, that demands a constant number of operations at the decryption stage; however, the access policy formats have to be expressed as a disjunctive normal form (DNF); therefore, the ciphertext size is proportional to the number of conjunctive clauses in the DNF.

Chase (2007) introduced a multi-authority ABE scheme in which several authorities cooperate to manage the attributes. Each authority manages a domain of attributes and distributes those attributes and secret keys to the users. The main issue affecting this scheme is that it is not practical to have one trusted central authority. An enhanced multi-authority ABE scheme was subsequently proposed by Chase and Chow (2009) that removes the trusted authority; however, in order to preserve user privacy, each authority has to assign at least one attribute to each user. Lewko and Waters (2011) provided a decentralized ABE scheme that does not require a trusted authority, but still maintains privacy. In their scheme, the access structure for any given user is only known by the sender; however, this decentralized mechanism is not suitable for cloud computing.

The scheme proposed by Yu et al. (2010a) exploits KP-ABE, by combining it with proxy re-encryption and lazy re-encryption. It simultaneously achieves fine-grainedness, scalability and data confidentiality for data access control. The data owner can delegate most of the computation tasks, such as user revocation, to the cloud server without disclosing any data to the untrusted cloud; however, by delegating these tasks, some user attributes and secret keys may leak into the cloud. Furthermore, the related ciphertext must be re-encrypted, allowing it to be revealed to non-revoked users. As such, we were able to discover the proxy re-encryption techniques applied in CP-ABE by Wang et al. (2010) and Yu et al. (2010b).

Li et al. (2010) provided a secure cloud storage scheme for health records in cloud computing by using Chase and Chow's multi-authority ABE scheme to divide users into different domains; however, this scheme is an isolated case and is not generally applied in cloud computing. Vimercati et al. (2010) presents a formal access control model on outsourced data. In their scheme, each file is encrypted with a symmetric key and each user is assigned a secret key; however, the complexity of operations of file creation and user grant/revocation is linear to the number of users, which makes this scheme unscalable. Moreover, Samarati and di Vimercati (2010) discusses some main privacy issues to be addressed in data outsourcing, ranging from data confidentiality to data utility. This paper is a good survey on data protection and privacy over outsourced database scenarios.

Wang et al. (2011) proposed a hierarchical fine-grained access control scheme that relies on Hierarchical IBE (Horwitz and Lynn, 2002) and CP-ABE. The architecture of this scheme is arranged in a hierarchical way with a root master and several domain masters to generate keys for users; however, because a large number of keys are required for each entity, the system is complicated.

3. Problem statement

3.1. System model

Our system model, as shown in Fig. 1, necessitates four parties in a network: The **data owner**; who has data stored in the cloud and depends on the cloud for data maintenance. Data owner can be enterprises or individual customers. The **data consumer**; who accesses the data shared by the data owner, downloads data of interest and decrypts it using his secret keys (for brevity, data consumers are referred to as users in this paper). The **cloud server (CS)**; provides a high-quality service utilizing a number of servers with considerable storage space and computation power. The **private key generator (PKG)**; is a trusted third party that computes corresponding private keys for users (Boneh and Franklin, 2001).

In our system, the PKG is only tasked with delivering public keys to the data owner and to generate and distribute corresponding private keys to the users, as described by Cocks (2001). Because there is no incentive to reveal data to the PKG, private data is kept separate from the PKG and no other security issues are transferred to the PKG. Secondly, the data

owner stores his data in a set of cloud servers which are running in a cooperated and distributed manner; therefore, the data owner can interact with the cloud servers to dynamically access and update his data via the CS provider (CSP). In addition, the CS has to always be online; whereas, data owners and consumers can be offline.

3.2. *Adversary model*

The adversary model considers most threats to cloud data confidentiality as malicious. In contrast, the CS in our model is semi-trusted (also known as passive), in that it behaves appropriately most of time; however, in certain situations an entity may arbitrarily deviate from the protocol specifications and the CS may try to acquire as much secret information as possible (De Capitani di Vimercati et al., 2007). We suggest that although the semi-trusted adversary model is weaker than the malicious model, it is often a more realistic model. The three types of threats can be categorized as follows:

1. Inner threats (from the CSP and users who might obtain unauthorized data), and outer threats (from unauthorized attackers and external adversaries beyond the system domain).
2. Active attacks (where unauthorized users inject malicious files into the cloud), and passive attacks (where unauthorized users eavesdrop on conversations between users and the cloud).
3. Collusion between the CSP and users (to access unauthorized data for the purpose of harvesting file contents).

In addition, each user can download a public/private key pair from the PKG, in which the public key is the user ID. The user ID may be a user identity card number, email address, etc. Attackers can easily obtain the user ID (public key) somewhere. Note that, in the system model, the communication channels between users and CS are secured under existing protocols, such as SSL.

3.3. *Security requirements*

With respect to secure data sharing and data access control in the cloud, the main goal of our model is to protect the cloud data from being accessed by inner intruders, including the cloud and from external attackers and unauthorized outer users. As such, our system has the following requirements:

1. **Fine-grained access control**: each user should only be able to access the data they are allowed to, with no access to unauthorized data.
2. **Collusion resistance**: users should not be able to collude with any other user, or the cloud, for the purpose of sharing their secret key to access unauthorized data.
3. **Backward secrecy**: the data access control policy should have the functionality to ensure that users are unable to access cloud data once their privileges have been revoked.

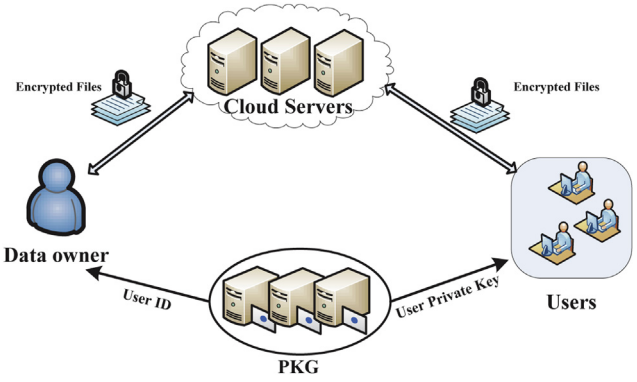

Fig. 1 – System model.

3.4. *Design goals*

The main design goals of our system are as follows: provide an adversary model, as previously described, with a secure, private and scalable policy for data sharing in cloud computing. As such, data owners are required to assign access structures to define which files users are allowed access, by generating unique key combinations for each attribute that are specific to each user; to protect users' privacy against the CS, by prohibiting the CS from learning the contents of user data or information on users' access privileges; to support dynamic user requests, such as addition and revocation of user access privileges; finally, the ensure the overheads of the service provided by the system are as light as possible.

4. The proposed scheme

In order to improve privacy and security for data sharing in cloud computing, we propose a scheme that combines CP-ABE (Bethencourt et al., 2007) and IBE (Shamir, 1985). Based on ABE, we choose two random exponents for every attribute, while the proposed scheme introduces a hash function that maps user IDs to group elements in the algorithm of key generation and decryption. Table 1 shows the symbols used in our scheme.

4.1. *Overview*

Our scheme is based on a bilinear map. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two cyclic groups of prime order q, and g_1 is the generator of group $\mathcal{G}_1$. A bilinear map $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ satisfies the following properties:

1. Bilinearity: for all $y, z \in \mathcal{G}_1$ and $a,b \in \mathbb{Z}_q$, where $\mathbb{Z}_q = \{0,1,2,\ldots q-1\}$, we have $e(y^a,z^b)=e(y,z)^{ab}$.
2. Computability: for any $y, z \in \mathcal{G}_1$, there is a polynomial time algorithm to compute $e(y,z) \in \mathcal{G}_2$.
3. Non-degeneracy: $e(g_1,g_1) \neq 1$.

Similar to CP-ABE (Waters, 2011), each file is described by a set of attributes. Determined by the set of attributes that a file has, an access tree is assigned to the file. The access tree is converted to a Linear Secret Sharing Scheme (LSSS) matrix (Beimel, 1996). Each user has a set of attributes given by the data owner and owns a unique ID regarded as his public key.

Table 1 – Symbols and their meanings.

Symbols	Meanings
i	Attribute i
U_u	User u
W	Total number of attributes in the system
M	LSSS Matrix
C	Ciphertext
$\mathcal{M}$	Message
UL	The system user list
ID_u	User U_u's ID
SK_u	User U_u's secret key
$SK[u]$	User U_u's private key
PK,SK	System public key and secret key
α_i,β_i	Secret key component for attribute i
I	Attribute set assigned to a data file
I_u	Attribute set assigned to user U_u
$\|I_u\|$	Number of attributes that U_u possesses
$\mathbb{A}$	User access structure
$sk_{i,u}$	Secret key corresponding to attribute i given to user U_u
H	Hash function, example SHA-1
$\delta_{O,X}$	The data owner's signature on message X
Enc_k^{sym}	Symmetric encryption under key k, example RC4 and CBC
Dnc_k^{sym}	Symmetric decryption under key k

For each attribute the user has, a key for the user ID is created. A user is able to decrypt the data in the cloud server if and only if he has a matched set of attributes. In addition, there exists a user list (UL) in the cloud to retain the authorized user *IDs* (public keys) in our system.

The proposed scheme elegantly integrates four randomized algorithms: ***System initialization***, ***Encryption***, ***Key generation***, ***Decryption*** to achieve effective, scalable and privacy-preserving cloud data sharing service. Fig. 2 describes a simplified workflow of the proposed scheme. At the initialization phase, the data owner uses the *System initialization* algorithm to generate system parameters for all system entities. The data owner then employs *Encryption* algorithm to encrypt files with "attributes" and uploads to the cloud. By the *Key generation* algorithm, the data owner generates secret keys for each user, and then delivers them to the users via the cloud server. At last, users use *Decryption* algorithm to decrypt ciphertext if their attribute set matches with the file attributes. In the following subsection, we elaborate the design in details.

4.2. *The proposed Scheme in detail*

Based on the system model, we provide the proposed scheme in detail. Our goal is to enable the authorized users to access and restore file correctly. Therefore, any other users outside the system will have no clue of any information about the file, even if they collude with the authorized users and the cloud. In this section, we first introduce the formats of user access policy and then present the four polynomial time algorithms.

4.2.1. *Formats of access policy*

Access policy can be expressed by an access tree $\mathbb{A}$ with attributes at leaves and logic gates e.g. *AND*($\wedge$), *OR*($\vee$) as intermediate nodes represented in ABE (Goyal et al., 2006). In our scheme, any access tree can be converted into a boolean formula. Any access tree $\mathbb{A}$ can be converted to a Linear Secret Sharing Scheme (LSSS) matrix M (Beimel, 1996). LSSS access structures are more general, and can be derived form representations as boolean formulas. There are standard techniques to convert any boolean formula into a corresponding LSSS matrix. The number of rows in the corresponding LSSS matrix will be same as the number of leaf nodes in the access tree. In LSSS, every piece is a vector over some finite field, and every set in the access structure reconstructs the secret using a linear combination of the coordinates of its pieces. Different from ABE, a message $\mathcal{M}$ is encrypted with an LSSS access structure (M,ρ) where ρ is a permutation function that maps rows of M to attributes in $\mathbb{A}$. Similar to Lewko and Waters (2011), the user who only has the secret keys for a subset of rows M_x of M such that (1,0,...0) is in the span of these rows can decrypt the message correctly. We will give an instance in Section 4.3 to demonstrate how this works. Please refer to Beimel (1996) for more details on Linear Secret Sharing Schemes.

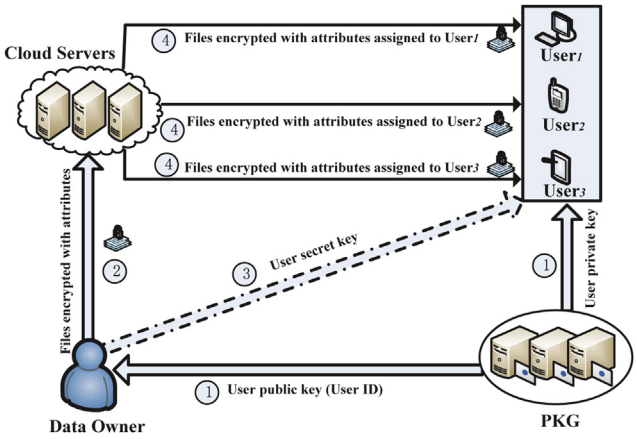

Fig. 2 – A simplified workflow of the proposed scheme.

4.2.2. *System initialization*

A data owner chooses a large prime q, two groups $\mathcal{G}_1, \mathcal{G}_2$ of order q, a map $e: \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ and a hash function $\mathcal{G}_1, \mathcal{G}_2$ which maps a user ID to a element of $\mathcal{G}_1$. Then the data owner defines a set of attributes W for sharing data files and selects two random exponents $\alpha_i,\beta_i \in \mathbb{Z}_q$ for each attribute in W. So the secret key SK for the system is

$$SK = \{\alpha_i, \beta_i, i \in W\}.$$

The public key PK for the system is published:

$$PK = \left\{e(g_1, g_1)^{\alpha_i}, g_1^{\beta_i}, i \in W\right\}.$$

4.2.3. *Encryption*

The data owner defines a set of attributes $I \in W$ for each data file. As described in Section 4.2.1, the formats of access policy can be represented as a $n \times l$ LSSS matrix M with a function ρ mapping its rows to attributes. The data owner processes the message $\mathcal{M}$ as follows:

- randomly select a seed $s \in \mathbb{Z}_q$ and a random vector $v \in \mathbb{Z}_q^1$ with the first entry as s. Let $\lambda_x = M_x \cdot v$ where M_x is row x of M.

- randomly select a vector $\omega \in \mathbb{Z}_q^1$ with the first entry as 0 and a seed $r_x \in \mathbb{Z}_q$. Let $\omega_x = M_x \cdot \omega$.
- encrypt the message $\mathcal{M}$ with (M,ρ) as follows:

$$C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x}, C_{2,x} = g_1^{r_x},$$
$$C_{1,x} = e(g_1,g_1)^{\lambda_x} e(g_1,g_1)^{\alpha_{\rho(x)} r_x}, C_0 = Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M}) \forall x,$$

where $\rho(x)$ is a permutation function mapping M_x to attribute i, and $Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$ is a symmetric encryption under key $e(g_1,g_1)^s$.

Finally, the data owner uploads the encryption file $C = \{\forall x,\{C_0,C_{1,x},C_{2,x},C_{3,x}\};(M,\rho)\}$ to the cloud servers.

4.2.4. *Key generation and distribution*

The data owner obtains user ID (ID_u) from PKG and assigns a set of attributes I_u for user U_u. Then the owner calculates the secret key component $sk_{i,u}$ for ID_u of attribute i belonging to user U_u:

$$sk_{i,u} = g_1^{\alpha_i} H(ID_u)^{\beta_i}.$$

The secret key for user U_u is $SK_u = \{sk_{i,u}, i \in I_u\}$. SK_u is encrypted by the user public key (ID_u) and delivered to the user via the cloud server, such that only that user (ID_u) can decrypt it using his private key.

4.2.5. *Decryption*

User U_u receives a ciphertext $C = \{\forall x,\{C_0,C_{1,x},C_{2,x},C_{3,x}\};(M,\rho)\}$ and $H(ID_u)$ from the cloud and selects constants $c_x \in \mathbb{Z}_q$ such that $\Sigma_x c_x M_x = (1,0,\ldots,0)$. The secret key of U_u is $\{sk_{i,u}, i \in I_u\}$. Then U_u calculates:

$$\begin{aligned}
&\Pi_x \left\{ C_{1,x} \cdot e(H(ID_u), C_{3,x}) / e(sk_{\rho(x),u}, C_{2,x}) \right\}^{c_x} \\
&= \Pi_x \left\{ e(g_1,g_1)^{\lambda_x} \cdot e(g_1^{\alpha_{\rho(x)}}, g_1^{r_x}) \cdot e\left(H(ID_u)^{\beta_{\rho(x)}}, g_1^{r_x}\right) \cdot \right. \\
&\left. e(H(ID_u), g_1^{\omega_x}) / e\left(g_1^{\alpha_{\rho(x)}} H(ID_u)^{\beta_{\rho(x)}}, g_1^{r_x}\right) \right\}^{c_x} \\
&= \Pi_x \left\{ e(g_1,g_1)^{\lambda_x} \cdot e(H(ID_u), g_1)^{\omega_x} \right\}^{c_x} \\
&= e(g_1,g_1)^{\Sigma_x \lambda_x c_x} \cdot e(H(ID_u), g_1)^{\Sigma_x \omega_x c_x} \\
&= e(g_1,g_1)^{v \cdot \Sigma_x c_x M_x} \cdot e(H(ID_u), g_1)^{\omega \cdot \Sigma_x c_x M_x} \\
&= e(g_1,g_1)^s \cdot e(H(ID_u), g_1)^0 \\
&= e(g_1,g_1)^s.
\end{aligned}$$

User U_u can obtain the message $\mathcal{M} = Dec^{sym}_{e(g_1,g_1)^s}(C_0)$.

Using the proposed scheme, the data owner encrypts files and stores them into the cloud, while the users decrypt the cipertext C using their own secret keys.

4.3. *How the scheme works in practice*

Considering the situation of a healthcare case, a medical center stores millions of healthcare records in the cloud. We consider a access structure of one record $\mathcal{M}$ as Fig. 3. The attribute set of this record is $I = \{i_1(Diabetes), i_2(Chinese), i_3(White), i_4(American)\}$.

The LSSS matrix M can be generated using the algorithm in Liu and Cao (2010). Thus, the matrix M for Fig. 3 is

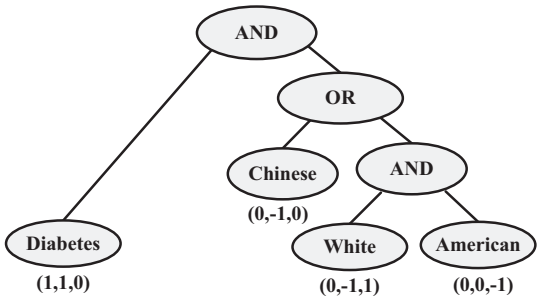

Fig. 3 – Access tree of one healthcare record for the medical center.

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} \quad (1)$$

Let the permutation function ρ be denoted as,

x	1	2	3	4
$\rho(x)$	i_1	i_2	i_3	i_4

The medical center encrypts this record $\mathcal{M}$ with (M,ρ) and system public key PK, and then sends the ciphertext $C = \{\{C_0,C_{1,x},C_{2,x},C_{3,x}\}_{x\in\{1,2,3,4\}};(M,\rho)\}$ with the hash function H to the cloud server. So a doctor who looks up records relating to "diabetes and Chinese" is able to access this record. However, if looking up "diabetes and American", he will be unable to access this record. Next, we will check these access policies.

Supposing there is a doctor U_2 (where we assume his ID is 2 for brevity) who looks up patients with "diabetes and white from American". He then is given attributes i_1,i_3 and i_4, so $I_2 = \{i_1,i_3,i_4\}$. Next he will be given the secret key $sk_2 = \{sk_{1,2},sk_{3,2},sk_{4,2}\}$ which is assigned by the medical center through the cloud servers. The doctor obtains $C = \{\{C_0,C_{1,x},C_{2,x},C_{3,x}\}_{x\in\{1,2,3,4\}};(M,\rho)\}$ and $H(ID_2)$ from the cloud servers. The doctor first looks for the common attributes associated with the record through the permutation function ρ, and gets common attributes i_1,i_3 and i_4. Then he finds corresponding vectors of attribute i_1,i_3 and i_4 are (1,1,0),(0,−1,1) and (0,0,−1) respectively in the LSSS matrix M. According to the decryption algorithm in Section 4.2, the doctor finds the linear combination of rows 1,3 and 4 to (1,0,0) as follows:

$$(1,1,0) + (0,-1,1) + (0,0,-1) = (1,0,0).$$

Then the doctor can use the decryption algorithm to calculate $e(g_1,g_1)^s$. The record $\mathcal{M}$ can be recovered once $e(g_1,g_1)^s$ is calculated.

Supposing there is a doctor who looks up records relating to "diabetes and American". The common attributes are i_1,i_4. Corresponding vectors of these attributes are (1,1,0) and (0,0,−1). We observe that there is no linear combination of rows 1 and 4 of matrix M to (1,0,0). Thus, the doctor can not calculate $e(g_1,g_1)^s$. Further on, he can not recover the record $\mathcal{M}$.

5. Dynamic operations

Our scheme is appropriate for some static application scenarios like libraries. However, there are many cloud data sharing scenarios where the cloud data is dynamically changing. Since the data owner stores the data into cloud server, rather than physically possing it, the dynamic data and user operations are quite challenging. The secret keys should not be known by the cloud server while processing the dynamics request. The dynamic operations such as file creation/deletion and user addition are processed as some existing work (Yu et al., 2010a). In addition, we give the process of user addition and modification of user attributes in detail. Moreover, the data owner needs to guarantee that all the operations should be processed faithfully by the cloud servers. To address these problems, we provide the dynamic data and user operations in this section. We also achieve modification of user attributes, which the prior works do not support.

5.1. File operations

From files' perspective, our system should consider the dynamic scenarios, where the data owner may create and delete data files to maintain storage correctly.

5.1.1. File creation

In cloud data sharing, there are cases when data owner uploads new data into the cloud servers. When the data owner wants to create a new file, he chooses a unique *ID* and defines the attribute set *I* for the new file. Then the data owner encrypts the file using the algorithm in Section 4.2 and uploads the encrypted file and LSSS matrix $C = \{\forall x,\{C_0,C_{1,x},C_{2,x},C_{3,x}\};(M,\rho)\}$ with the his signature $(C,\delta_{O,C})$ to the cloud. If verifying the signature $(C,\delta_{O,C})$ correctly, the cloud stores the new file. After uploading the encrypted file into the cloud, the data owner can go offline at any time.

5.1.2. File deletion

Sometimes, some outdated cloud data should be deleted. The delete operation we consider here is straightforward. Only the data owner has the privilege to delete his stored file. When the data owner wants to delete an outdated file, he sends the file *ID* and his signature to the cloud. After verifying the signature on this file *ID*, the cloud deletes the outdated file.

5.2. User operations

From users' perspective, to preserve the cloud data security, new users will join and outdated users need to be revoked.

5.2.1. User addition

From the user's perspective, there are some new users who want to join the system to access the shared data. When a new user U_u joins the system, the data owner first obtains the user's *ID* (ID_u) from the PKG, assigns the attribute set I_u and calculates the corresponding secret key for this new user. The data owner then sends the secret key and his signature to the cloud server. After verifying the signature, the cloud sends the secret key and related secret information to the new joining user. The user decrypts the message to get his secret key in the system. The data owner first obtains the new user *ID* (ID_u) from PKG, assigns a set of attributes I_u for U_u and calculates the secret key $SK_u = \{sk_{i,u}, i \in I_u\}$ for ID_u. Then it encrypts the secret key, attribute set and the corresponding hash value $H(ID_u)$

$$\left(I_u, SK_u, H(ID_u), \delta_{O,(I_u,SK_u,H(ID_u))}\right)$$

with user's *ID*, denoting as *D*. Finally it sends ciphertext *D* and user's *ID* $(D,ID_u,\delta_{O,(D,IDu)})$ to the cloud. After receiving the message from the data owner, the CS verifies the signature $\delta_{O,(D,IDu)}$. If failed in signature verification, the CS deletes the received ciphertext. Otherwise, the CS saves user's *ID* (ID_u) in *UL* and sends the ciphertext *D* to the joining user. The joining user first obtains his private key *SK*[*u*] from PKG. After decrypting the ciphertext *D* using his private *SK*[*u*], he verifies the signature $\delta_{O,(Iu,SKu,H(IDu))}$. Finally, the joining user accepts $(I_u,SK_u,H(ID_u))$ as his access attribute set, secret key and user *ID* corresponding hash value. After receiving the secret keys, the newly joined user can access the matched files correctly. The cloud server only obtains the user's *ID* and system public key but no secret keys. Thus, privacy and security can be achieved.

5.2.2. User revocation

In some cases, the data owner may revoke some users' access privileges. After being revoked, these users are not allowed to access the cloud data anymore. In some early works, the data owner updates the secret keys corresponding to the attributes that the revoked user possesses. Then the data owner re-encrypts the related files and distributes the new keys to the non-revoked users via the cloud server. Although it is also suitable for our scheme, it discloses users' access privileges to the cloud and brings more computation overhead. We have a more optimizing method to deal with user revocation. In our scheme, the data owner only re-encrypts part of the ciphertext and thus there is no need to update the corresponding secret keys. When there exists a user to be revoked, the data owner first determines the set of attributes I_u which user U_u possess. Then, he randomly chooses a new vector of $(v)_{new} \in \mathbb{Z}_q^1$. Now the new first entry of vector $(v)_{new}$ is $(s)_{new}$. A new $(\lambda_x)_{new} = M_x \cdot (v)_{new}$ is calculated for each LSSS matrix row *x* corresponding to attributes belong to I_u. The data owner recalculates the new values of $(C_{1,x})_{new}$ and $(C_0)_{new}$ as

$$(C_{1,x})_{new} = e(g_1,g_1)^{(\lambda_x)_{new}} e(g_1,g_1)^{\alpha_{\rho(x)} r_x}, (C_0)_{new} = Enc^{sym}_{e(g_1,g_1)^{(s)_{new}}}(\mathcal{M}).$$

Finally he sends file *ID f* and user's *ID* along with the new encrypted file,

$$\left(f, ID_u, (C_0)_{new}, (C_{1,x})_{new}, \delta_{O,\left(f,ID_u,(C_0)_{new},(C_{1,x})_{new}\right)}\right)$$

to the cloud. After verifying the signature $\delta_{O,(f,ID_u,(C_0)_{new},(C_{1,x})_{new})}$, the CS deletes the old encrypted file and ID_u from the *UL*. It stores the new received one on the base of file *ID*. Since we do not update the secret keys for non-revoked users, they access the cloud data just as given in Section 4.2. To prevent the revoked user eavesdrop the communication, the cloud can use non-revoked users' public key to encrypt the new encrypted file. In the stage of decryption, only the user obtains the exact $C_{1,x}$ can decrypt the message $\mathcal{M}$, which can prevent the revoked user from accessing the cloud file.

5.3. Change of user attributes

In other scenarios, when the data owner wants to adjust some users' privileges, he needs to change the user's attribute set. Suppose the data owner changes the attributes for a specific user. In some existing works using access trees, the related data needs to be re-encrypted and the keys for the other's have to be re-generated and distributed. However, we can see that our scheme does not need to reconstruct the secret keys. In certain situations, if the data owner wants to revoke some of attributes about the specific user, he first determines the set of revoked attributes I_u which user U_u possess. Then, the data owner will reselect $v_{new} \in \mathbb{Z}_q^1$ and recalculates the new values of $(C_{1,x})_{new}$ and $(C_0)_{new}$ as described in *User Revocation*. The process is the same as user revocation operations. For each row x corresponding to attributes in I_u, $C_{1,x}$ is recalculated and the new $(C_{1,x})_{new}$ is not transmitted to user U_u. Therefore, user U_u cannot decrypt unauthorized data. Furthermore, if a specific user wants to add some permissions, the data owner just generates the new keys corresponding to these attributes and delivers to the user via cloud as *User Addition*. The user can use the new keys to decrypt the corresponding ciphertext.

6. Security analysis

We analyze the proposed scheme in term of security. The security analysis focuses on the security requirements defined in Section 3. In our scheme, we assign flexible and different access privileges for each user to achieve fine-grained access control. Meanwhile, our scheme achieves fully collusion secure which is important when several users collude and share their secret keys to access the unauthorized data. Our scheme can also achieve user access privilege confidentiality. In this section, we first show that our scheme is secure under the generic bilinear group model in the random oracle model. Then, the security requirements are focused on. At last, we present user access privilege confidentiality to realize users' privacy.

6.1. Security

We first define the security of the proposed scheme in the sense of semantic security. Give a ciphertext to the adversary and he learns nothing about the corresponding plaintext. The security of the proposed scheme is defined as a game under the aforementioned adversary model. As mentioned in Section 3.3, there are two main threats in our system, i.e., inner threats initiated by the "curious" CSP and some authorized users, and outer threats initiated by some unauthorized users. Meanwhile, the threat has both active and passive capabilities, which may collude to obtain the cloud data, and may eavesdrop on the communication traffic. We present the security game between a challenger and an adversary as follows:

Setup: The challenger runs the system initialization algorithm to generate the system public/secret key in Section 4.2. It gives the public parameters *PK* to the adversary and keeps the secret key *SK* to itself.

Phase 1: The adversary is allowed to issue secret key extraction queries by submitting pairs (i, ID) to the challenger, where i is an arbitrary attribute and ID is an identity. The challenger runs the key generation algorithm to generate the corresponding key $sk_{i,ID}$ and sends it to the adversary. The quires may be requested adaptively.

Challenge: Once the adversary determines Phase 1 finished, it declares two equal length messages $\mathcal{M}_0$, $\mathcal{M}_1$ and an access matrix (M,ρ) which are supposed to be challenged. The access matrix cannot be satisfied by any of the queried attributes in Phase 1. The challenger picks a random coin $b \in \{0,1\}$ and encrypts $\mathcal{M}_b$ under access matrix (M,ρ). It gives the ciphertext to the adversary.

Phase 2: The adversary is allowed to issue additional key queries (i, ID), with the added restriction that none of these satisfy (M,ρ). The response of the challenger is as in Phase 1.

Guess: The adversary outputs a guess b' for b and wins the game if $b' = b$. The advantage of an adversary in breaking our scheme is defined as $|Pr[b' = b] - 1/2|$.

Definition 6.1. *We say that the proposed scheme is secure if all polynomial time adversaries have at most a negligible advantage in this security game.*

Theorem 6.1. *The proposed scheme is secure in the generic bilinear group model, modeling H as a random oracle.*

Proof. The generic bilinear group model is used in Boneh et al. (2005) and Bethencourt et al. (2007). Security in this model assures that an adversary cannot crack the proposed scheme with only black-box access to the group operations and the hash function H.

In the above security game, the adversary must distinguish between $C_0 = Enc^{sym}_{e(g_1,g_1)^s}(M_0)$ and $C_0 = Enc^{sym}_{e(g_1,g_1)^s}(M_1)$. Here we consider a modified game, where the adversary must distinguish between $C_0 = e(g_1,g_1)^s$ or $C_0 = e(g_1,g_1)^t$ where t is chosen uniformly randomly from $\mathbb{Z}_q$. This is justified by a simple hybrid argument in Bethencourt et al. (2007). We first simulate the modified game where C_0 is set to be $e(g_1,g_1)^t$.

The challenger runs the initialization algorithm and gives g_1 to the adversary. Then it randomly chooses exponents $\alpha_i, \beta_i \in \mathbb{Z}_q$ for the attributes $i \in W$, and queries the group oracles for each $e(g_1,g_1)^{\alpha_i}, g_1^{\beta_i}$ and gives these to the adversary.

When the adversary requests $H(ID)$ of some ID for the first time, the challenger chooses a random value $h_{ID} \in \mathbb{Z}_q$ and queries the group oracle for $g_1^{h_{ID}}$ and gives this to the adversary as $H(ID)$. The challenger stores this value.

When the adversary issues a key $sk_{i,ID}$, the challenger calculates $g_1^{\alpha_i} H(ID)^{\beta_i}$ using the group oracle and gives this to the adversary. If $H(ID)$ has not been requested before, it is computed as above.

When the adversary specifies an access matrix (M,ρ) for the challenge ciphertext and gives the challenger with the $e(g_1,g_1)^{\alpha_i}, g_1^{\beta_i}$ values that appear in the image of ρ on the rows of M, the challenger checks these are valid elements by querying the group oracles. The challenger now generates the challenge ciphertext. It runs the encryption algorithm to produce the ciphertext. Using the group oracles, the ciphertext is different from Section 4.2:

$C_{3,x} = g_1^{\beta_{\rho(x)} r_x} g_1^{\omega_x}, C_{2,x} = g_1^{r_x},$
$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x}, C_0 = e(g_1, g_1)^t \forall x,$

where t is a random value from $\mathbb{Z}_q$. The challenger gives the ciphertext to the adversary.

We argue that the adversary's view in the modified game is identically distributes if C_0 had been set to $e(g_1,g_1)^s$ instead of $e(g_1,g_1)^t$. This shows that the adversary obtains a negligible advantage in the modified game and hence obtains a negligible advantage in the real security game.

We consider an event that each of the adversary's queries to the group oracles have input values that were given to the attacker or were received from the oracles in response to previous queries. This will happen with high probability. Under this situation, we take each of the queries as a multivariate polynomial with the variables h_{ID},α_i,β_i,r_x,ω_x,λ_x,t, where ID ranges over the allowed identities and x ranges over the rows of the access matrix. We further consider the event that the random assignment to the variables h_{ID},α_i,β_i,s,t,r_x receives two different answers of two query polynomials.

Since t only appears in C_0, the adversary only can make queries involving t which are the form ct plus other terms, where c is a constant. If the adversary makes two queries f and f' which are unequal polynomials but become the same when we replace s with t, the adversary's view still will differ when $s = t$. Thus, $f - f' = cs - ct$ for some c. We now show that the adversary cannot make a query of the form cs. In the modified game, the adversary only can form queries which are linear combinations of 1,t. Furthermore, the adversary knows the values of α_i,β_i for attribute i. We recall that $\lambda_x = M_x \cdot v$ where vector v's first entry is s and this is the only appearances of s. To make a query of the form of cs, the adversary should select constants b_x such that $\Sigma_x \lambda_x = cs$ and form:

$$\Sigma_x b_x \left(\lambda_x + \alpha_{\rho(x)} r_x\right).$$

For the term $b_x \alpha_{\rho(x)} r_x$, the adversary knows the value $\alpha_{\rho(x)}$. To cancel this term, the adversary can form the term $-b_x \alpha_{\rho(x)} r_x$, so the adversary cancels this term by using:

$$-b_x \left(\alpha_{\rho(x)} r_x + h_{ID} \beta_{\rho(x)} r_x\right).$$

This brings a new term of $-b_x h_{ID} \beta_{\rho(x)} r_x$ to be canceled. The adversary only requests a key for the attribute, identity pair ($\rho(x)$,ID) to access to the term $\alpha_{\rho(x)} r_x + h_{ID} \beta_{\rho(x)} r_x$. Therefore, the extra term $-b_x h_{ID} \beta_{\rho(x)} r_x$ can be canceled by using:

$$b_x \left(h_{ID} \beta_{\rho(x)} r_x + h_{ID} \omega_x\right).$$

And this brings an additional term $b_x h_{ID} \omega_x$. The collection of these terms for identity ID can cancel if the vector (1,0,…0) is in the span of the rows M_x of M or the adversary can attain the key for attribute, identity pair ($\rho(x)$,ID). If this is established for the ID, then the adversary issued a collection of keys for the same identity ID which has the ability to decrypt the challenge ciphertext. Thus, the adversary has broken the security game rules.

As a result, we have shown that the adversary cannot make a query of the form cs for a constant c. In short, when t is random, the adversary's view is the same as when $t = s$. Hence, this shows that the adversary obtains a negligible advantage in the security game. Theorem 6.1 is true.

According to Theorem 6.1, we can conclude that the proposed scheme is secure in the generic bilinear group model, modeling H as a random oracle. The adversary cannot recover the constant c in our scheme, so it cannot obtain the plaintext. In this sense, the proposed scheme is secure.

6.2. Fine-grained access control

In our scheme, each user receives a flexible access structure from the data owner. Each user U_u has been assigned a set of attributes for the data owner. Suppose a file has an attribute $i \in I$, so it has a corresponding row r_b in the LSSS matrix. However, if the user U_u does not have the attribute i, he can not receive the secret key $sk_{i,u}$ for attribute i. In addition, in the decryption stage, as U_u cannot find the corresponding c_x of row r_x to satisfy $\Sigma_x c_x M_x = (1,0,\ldots,0)$, the decryption procedure will fail. Therefore, a user who does not have the attribute i cannot calculate $e(g_1,g_1)^s$. Thus, the user cannot decrypt the unauthorized message. Our scheme only discloses decryption keys to authorized users, thus unauthorized users and the cloud server cannot decrypt. For this reason, our scheme can help the data owner to realize fine-grained access control of the cloud data.

6.3. Fully collusion secure

We show that our scheme is fully collusion secure when users collude. We have the following theorem.

Theorem 6.2. *Two or more users with different identities cannot construct $e(g_1,g_1)^s$, even if they collude and combine their keys.*

Proof. The user ID is "tied" together with the given attributes so that users cannot combine the attributes of others in decryption. In encryption algorithm, the file $\mathcal{M}$ is blinded with $e(g_1,g_1)^s$. The value s is split into the vector λ_x and value 0 is split into the vector ω_x. The user who wants to obtain the file $\mathcal{M}$ must recover $e(g_1,g_1)^s$ by pairing keys for attributes and ID pairs. To achieve this, the user must introduce the term $e(H(ID),g_1)^{\omega_x}$. If the user has the matched set of keys, this term will be canceled in the decryption process. Otherwise, the term can not be canceled. If two or more users with different IDs attempt to collude, the terms $e(H(ID),g_1)^{\omega_x}$ will not cancel each other because the terms for each user are different. Hence, Theorem 6.2 holds true.

In our scheme, authorized users receive the ciphertext $C = \{\forall x, \{C_0, C_{1,x}, C_{2,x}, C_{3,x}\}; (M,\rho)\}$. The message $\mathcal{M}$ is encrypted in the form of $C_0 = Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$. According to Theorem 6.2, $e(g_1,g_1)^s$ cannot be constructed to recover message $\mathcal{M}$. Moreover, since the cloud and users do not have the secret keys for unauthorized data, they are unaware of any information in regards to the unauthorized data, even if they collude each other. Therefore, our scheme achieves fully collusion secure.

6.4. Backward secrecy

Backward secrecy can be realized in the proposed scheme. That is, the user who is revoked cannot decrypt data which was previously able to be accessed. As described in Section

5, our scheme will update part of the ciphertext ($C_{1,x}$) after some legitimate users are revoked. Since $C_{1,x}$, which depends on the random s, is recalculated and not sent to the revoked user, the revoked user is not able to recover $e(g_1,g_1)^s$ and decrypt the message. Therefore, our scheme is with backward secrecy. However, in the existing works, the data owner need to re-distribute keys for non-revoked users to guarantee backward secrecy. The redistribution will disclose users' secret key to the cloud and add additional communication cost.

6.5. User access privilege confidentiality

The proposed scheme does not disclose any attribute of a user attribute set to the cloud servers. In our key generation algorithm, users' access structures and secret keys are assigned by the data owner. The cloud stores and delivers ciphertext to cloud users. The secret keys are encrypted by users' public keys and then delivers to users. Thus, the cloud has no clue about users' secret keys and does not possess any $sk_{i,u}$. Therefore, the cloud cannot derive any user's access privilege information so that users' privacy are protected against the cloud. Moreover, in our user revocation and attribute change schemes, we need not to update the non-revoked users' secret keys. The cloud will transmit the new $C_{1,x}$ and C_0 to non-revoked users. The cloud still cannot obtain any user's privilege information. In contrast, Yu's scheme discloses leaf nodes information to the cloud. Only the interior nodes are unknown to the cloud. In addition, the cloud also knows part of the user's secret keys. Thus the more legitimate users are revoked, the more secret keys the cloud knows. This cannot achieve fully privacy-preserving policy in cloud computing.

According to the above analysis, we can see that the proposed scheme can achieve the desired security requirements, i.e., fine-grained access control, collusion resistance, and backward secrecy. Furthermore, the data owner and user's identity is public in our scheme, but it is supposed to be hidden under some circumstances. Our scheme is a generalized scheme that can incorporate other attribute based signature scheme (Shaniqng and Yingpei, 2008). Besides, during the encryption in our scheme, the cloud may learn the access structure $\mathbb{A}$ about the shared file through the LSSS matrix M. We can hide the access structure by using some converting algorithm which is irreversible (Liu and Cao, 2010). To prevent the leak of any sensitive information to the cloud, we do not delegate computation tasks to the untrusted cloud server in our scheme. Though it might increase some local computation, it does not significantly augment the overhead of computation and of communication.

7. Performance analysis

In this section, the performance of our scheme is analyzed by comparing with other data sharing schemes that rely on KP-ABE like Yu et al. (2010a). We first evaluate the computation and communication overhead, and then give the detailed about the ciphertext size in the proposed scheme.

7.1. Computation complexity

We analyse the computation overhead of the proposed scheme according to the encryption and decryption algorithms in this section. In the proposed scheme, the main computation operations involved in encryption and decryption algorithms are pairing (calculate $e(g_1,g_1)$) and scalar multiplication. We recall that the scheme chooses elliptic curve groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of order q. The ciphertext of the proposed scheme is $C=\{\forall x,\{C_0,C_{1,x},C_{2,x},C_{3,x}\}\}$. Pairing is the most expensive operation. For each different file, however, data owner and users only need to calculate $e(g_1,g_1)$ once in the beginning. Since both the proposed scheme and KP-ABE-based schemes have the same numbers of pairing operation, we do not involve in pairing operation overhead when computation complexity of the proposed scheme compares with the KP-ABE-based schemes. In the computation complexity analysis, we only take into account scalar multiplication operation. During encrypting, all encryption operations are at the data owner side. The data owner needs to do two scalar multiplications to calculate $C_{1,x}$, one scalar multiplication for $C_{2,x}$ ($C_{2,x}=g_1^{r_x}$), and one for $C_{3,x}$ for each row x in LSSS matrix. Therefore, the data owner needs at most $4|I|$ scalar multiplications. The computation complexity of data owner converting the access structure to an LSSS matrix is $\mathcal{O}(|I|)$ where $|I|$ is the number of attributes about the access structure. Thus, the computation complexity of encryption is $\mathcal{O}(|I|)$. In the decryption stage, the decryption operation is similar only for users. To recover ciphertext, the user needs at most another $|I|$ scalar multiplications to calculate $\prod_x\left\{e(g_1,g_1)^{\lambda_x}e(H(ID),g_1)^{\omega_x}\right\}$, so the time complexity is also $\mathcal{O}(|I|)$. The computation complexity of our scheme and KP-ABE-based schemes is given in Table 2. From Table 2, we notice that the computation complexity of encryption performed by the data owner in our scheme is the same with KP-ABE-based schemes. In addition, the number of N in KP-ABE-based schemes is bigger than $|I|$ most of the time, so our scheme consumes less computational cost in decryption stage. Therefore, our scheme achieves higher performance in privacy and security without increasing computational complexity.

We also conduct a thorough experimental evaluation about the time cost of the proposed scheme and KP-ABE-based schemes. The whole experiment system is implemented by Python language on a Windows 7 machine with Core 2 Duo CPU running at 2.0 GHz. All results are the average of 100 trials. First we test the speed of encryption. In our scheme, the calculation of C_0 is based on $C_{1,x},C_{2,x},C_{3,x}$. The encryption speed of our scheme and KP-ABE-based schemes is given in Fig. 4. Fig. 4 plots the overhead to calculate ciphertext versus the number of attributes $|W|$. From Fig. 4, we can see the encryption cost increases linearly with the attributes $|W|$ both in our scheme and KP-ABE-based schemes. This is consistent with the above computation analysis ($\mathcal{O}(|I|)$). However, with the increase of attributes, our scheme takes less time cost than KP-ABE-based schemes. Fig. 7 plots the performance of two common symmetric encryptions, i.e., 128-bit RC4 and 128-bit AES CBC, which are needed to calculate $C_0=Enc^{sym}_{e(g_1,g_1)^s}(\mathcal{M})$. From Fig. 7, we can see that it is effective to compute C_0, e.g. the time to encrypt a 10 MB file using 128-bit RC4 and 128-bit AES CBC approaches to 35 ms and 105 ms

Table 2 – Computation complexity required in KP-ABE-based schemes and our scheme.

Scheme	Encryption (Data owner)	Decryption (User)
KP-ABE-based	$\mathcal{O}(\|I\|)$	$\mathcal{O}(max(\|I\|,N))$
Ours	$\mathcal{O}(\|I\|)$	$\mathcal{O}(\|I\|)$

respectively, which is an ideal result. The overhead of key generation and decryption is shown in Figs. 5 and 6. In these tests, we assume all attributes should be involved in the key generation and decryption. Fig. 5 plots the overhead to calculate keys versus the number of attributes |W|. As we can see, the overhead also grows linearly with |W| in our scheme. Nevertheless, in KP-ABE-based schemes, it grows exponentially with |W|. Fig. 6 plots the speed to recover ciphertext. Fig. 6 also plots that the decryption cost grows linearly with |W| both in our scheme and KP-ABE-based schemes. However, with the increase of attributes, our scheme takes less time cost than KP-ABE-based schemes as well. Moreover, we find that it is cheaper than encryption in our scheme. The reason is because decryption takes less power operations. The results of our experiments show our scheme is light weighted and efficient to be applied in practice.

7.2. *Communication cost*

In our scheme, the communication cost is mainly attributable to the encrypted data transmission. After encryption, the following information is sent by the data owner along with the encrypted data to the cloud: Value of matrix M which requires $|I|^2$ bits, value of permutation function ρ requiring $log|I|$ bits, value of C_0,$C_{1,x}$,$C_{2,x}$ and $C_{3,x}$ for every x, $\log|\mathcal{G}_2| + |I|\log|\mathcal{G}_2| + 2|I|\log|\mathcal{G}_1|$, and value of $H(ID)$ which requires $log|\mathcal{G}_1|$ bits. Thus, the communication cost is given by $\left|I\right|^2 + \log\left|I\right| + (2|I|+1)\log\left|\mathcal{G}_1\right| + (|I|+1)\log\left|\mathcal{G}_2\right|$+Data. Table 3 shows the communication expenses comparison between

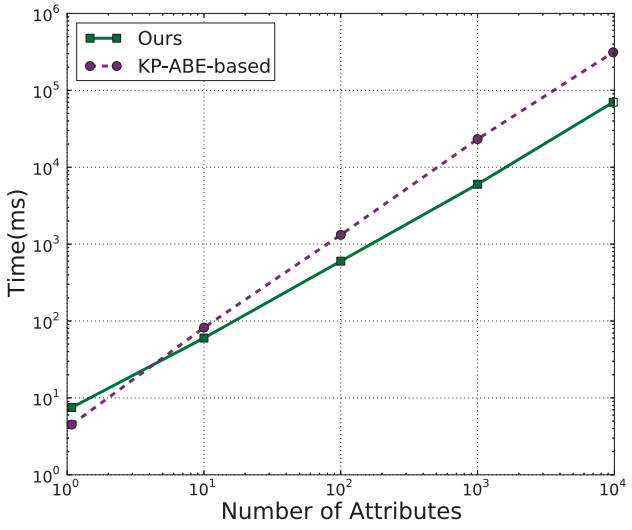

Fig. 4 – The overhead of encryption speed in our scheme and KP-ABE-based schemes.

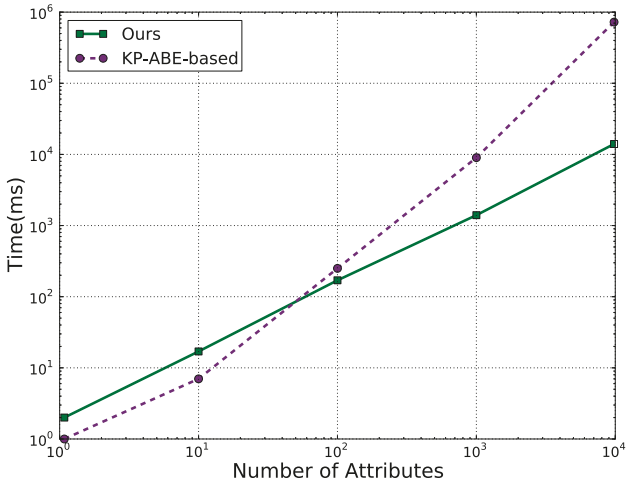

Fig. 5 – The overhead of key generation algorithms in our scheme and KP-ABE-based schemes.

our scheme and KP-ABE-based schemes. We can see that our scheme communication cost is a little more. However, in practice, a file is described by just a few attributes, i.e., $|I|$ is small in general cases. For example, in Section 4.3, a record just is described by diabetes, Chinese, White and American, i.e., $|I| = 4$. In addition, even though the order of cyclic group $\mathcal{G}$ is large, $\log|\mathcal{G}|$ bits is far less than the file size (Data). For example, the order of $\mathcal{G}$ is equal to 10^{10}, $\log|\mathcal{G}|$ bits is just near to 30 bits. Therefore, the main communication costs will depend on the file size. In other words, we actually can ignore the extra communication cost.

7.3. *Cost of revocation operation*

When user revocation is required, the ciphertext needs to be re-encrypted in our scheme. The data owner will choose a new seed s randomly and recalculate C_0 and $C_{1,x}$. Suppose the revoked user is U_u. Pairing($e(g_1,g_1)$) has been calculated, so the data owner only needs one scalar multiplication to recalculate C_0. For each attribute $x \in I_u$, there are another two scalar multiplications to update $C_{1,x}$. Therefore, there are totally $2|I_u| + 1$ scalar multiplications to re-encrypt the ciphertext by

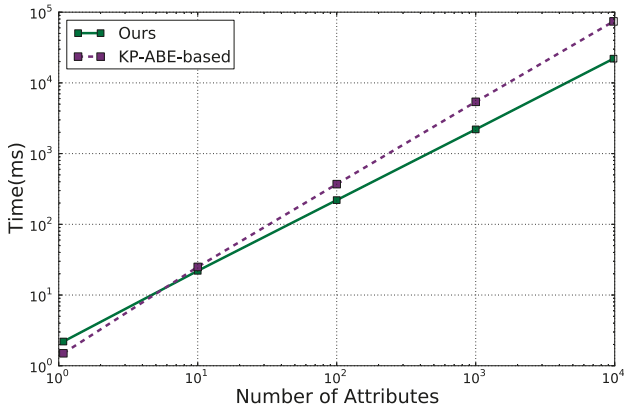

Fig. 6 – The overhead of decryption algorithms in our scheme and KP-ABE-based schemes.

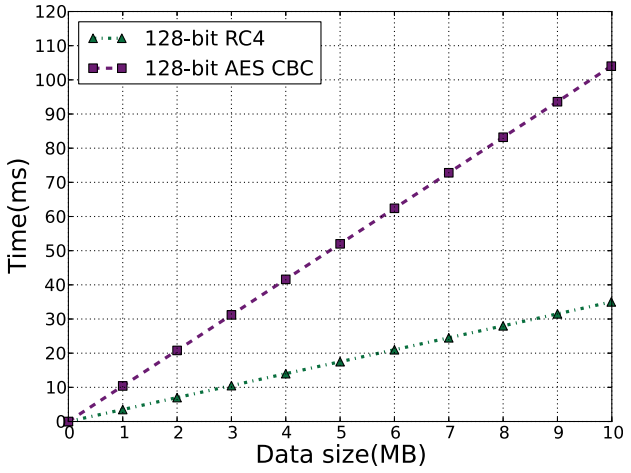

Fig. 7 – The cost of symmetric encryption algorithms.

the data owner. For the non-revoked users, they do not need to do any computation. Moreover, the data owner needs to send the new ciphertext to the cloud, while the cloud just replaces the outdated ciphertext and does not need to transfer it to the non-revoked users, so the additional communication costs is $(|I|+1)\log|G_2|$+Data. In the existing works, when revocation happens, the data owner needs to re-encrypt the related ciphertext and issue the new keys to those non-revoked users. Compared with our scheme, this brings an abundance of additional computation and communication overhead. Our scheme can accomplish this dynamic request with lightweight computation complexity.

7.4. *Ciphertext size*

As described in Section 4.2, the ciphertext is composed of four parts: C_0,$C_{1,x}$,$C_{2,x}$,$C_{3,x}$, so the size of the ciphertext is $(|I|+1)\log|\mathcal{G}_2|+2|I|\log|\mathcal{G}_1|+Data$. We compare the ciphertext size of our scheme with other KP-ABE-based schemes in Table 4. As discussed in Section 7.2, $|I|$ and $\log|\mathcal{G}|$ are far less than the file size (Data), so the difference between our scheme and KP-ABE-based schemes is also negligible.

8. Conclusion

In this paper, we present a privacy-preserving and secure data sharing scheme in cloud computing by exploiting CP-ABE and combining it with technique of IBE. The proposed scheme ensures fine-grained data access control, backward secrecy and security against collusion of users with the cloud and supports user addition, revocation and attribute modifications which are not provided by current works. Moreover, our scheme does not disclose any attribute of users to the cloud so that keeps the privacy of the users away from the cloud. Security analysis show that the proposed scheme is semantical security in the generic bilinear group model, modeling H as a random oracle. In addition, we evaluate the performance of the proposed scheme about computation complexity, communication cost and ciphertext size. The result shows that the proposed scheme is low overhead and highly efficient. Following the current research, we will implement the proposed privacy-preserving and effective cloud data sharing service in a real CSP platform for future work.

Table 3 – Communication costs in KP-ABE-based schemes and our scheme.

Scheme	Communication costs
KP-ABE-based	$\|I\|+2\log\|I\|+(\|I\|+1)\log\|\mathcal{G}_1\|+\log\|\mathcal{G}_2\|$+Data
Ours	$\|I\|^2+log\|I\|+(2\|I\|+1)\log\|\mathcal{G}_1\|+(\|I\|+1)\log\|G_2\|$+Data

Table 4 – Ciphertext size in KP-ABE-based schemes and our scheme.

Scheme	Ciphertext size
KP-ABE-based	$\|I\|+\|I\|log\|\mathcal{G}_1\|+log\|\mathcal{G}_2\|$+Data
Ours	$2\|I\|log\|\mathcal{G}_1\|+\|I\|log\|\mathcal{G}_2\|$+Data

Acknowledgments

This work was supported in part by the National Basic Research Program of China under Grants 2012CB316100, 2013CB338004, and the Research Fund for the Doctoral Program of Higher Education of China (NO. 20100073110016 and NO. 20120073120034).

REFERENCES

Amazon web services. Retrieved online from, http://aws.amazon.com/.

Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, et al. Above the clouds: a berkeley view of cloud computing [Technical report]. Berkeley: EECS Department, University of California; 2009 [Tech. Rep. UCB/EECS-2009-28].

Arrington M. Gmail disaster: reports of mass email deletions [Retrieved online on 28 December 2006 from], http://www.techcrunch.com/2006/12/28/gmail-disasterreports-ofmass-email-deletions/; 2006.

Ateniese G, Di Pietro R, Mancini L, Tsudik G. Scalable and efficient provable data possession. In: Proceedings of the 4th international conference on security and privacy in communication Netowrks (SecureComm). ACM; 2008. pp. 901–10.

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

Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: IEEE Symposium on security and privacy (SP). IEEE; 2007. pp. 321–34.

Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Advances in Cryptology–CRYPTO. Springer; 2001. pp. 213–29.

Boneh D, Boyen X, Goh EJ. Hierarchical identity based encryption with constant size ciphertext. In: Advances in Cryptology–EUROCRYPT. Springer; 2005. pp. 562–78.

Chase M. Multi-authority attribute based encryption. In: Theory of cryptography (TCC). Springer; 2007. pp. 515–34.
Chase M, Chow S. Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 16th ACM conference on computer and communications security (CCS). ACM; 2009. pp. 121–30.
Cocks C. An identity based encryption scheme based on quadratic residues. In: Cryptography and coding (IMACC). Springer; 2001. pp. 360–3.
De Capitani di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Samarati P. Over-encryption: management of access control evolution on outsourced data. In: Proceedings of the 33rd international conference on very large data bases (VLDB). VLDB endowment; 2007. pp. 123–34.
De Capitani di Vimercati S, Foresti S, Jajodia S, Paraboschi S, Samarati P. Encryption policies for regulating access to outsourced data. ACM Trans Database Syst (TODS) 2010;35(2):12:1–12:46.
Erway C, Küpçü A, Papamanthou C, Tamassia R. Dynamic provable data possession. In: Proceedings of the 16th ACM conference on computer and communications security (CCS). ACM; 2009. pp. 213–22.
Google app engine. Retrieved online from, http://code.google.com/appengine/.
Goyal V, Pandy O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security (CCS). ACM; 2006. pp. 89–98.
Horwitz J, Lynn B. Toward hierarchical identity-based encryption. In: Advances in Cryptology–EUROCRYPT. Springer; 2002. pp. 466–81.
Lewko A, Waters B. Decentralizing attribute-based encryption. In: Advances in Cryptology–EUROCRYPT. Springer; 2011. pp. 568–88.
Li M, Yu S, Ren K, Lou W. Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings. In: International conference security and privacy in communication networks (SecureComm). Springer; 2010. pp. 89–106.
Liu Z, Cao Z. On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption; 2010 [Technical Report; Cryptology ePrint Archive, Report 2010/374].
Microsoft azure. Retrieved online from, http://www.microsoft.com/azure.
Müller S, Katzenbeisser S, Eckert C. Distributed attribute-based encryption. In: Information security and cryptology (ICISC). Springer; 2009. pp. 20–36.
Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM conference on computer and communications security (CCS). ACM; 2007. pp. 195–203.
Sahai A, Waters B. Fuzzy identity-based encryption. In: Advances in Cryptology–EUROCRYPT 2005. Springer; 2005. pp. 557–73.
Samarati P, De Capitani di Vimercati S. Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM Symposium on information, computer and communications security (ASIACCS). ACM; 2010. pp. 1–14.
Shah M, Baker M, Mogul J, Swaminathan R. Auditing to keep online storage services honest. In: Proceedings of the 11th USENIX workshop on hot topics in operating systems. USENIX Association; 2007. pp. 1–6.
Shamir A. Identity-based cryptosystems and signature schemes. In: Advances in cryptology. Springer; 1985. pp. 47–53.
Shaniqng G, Yingpei Z. Attribute-based signature scheme. In: International conference on information security and assurance (ISA). IEEE; 2008. pp. 509–11.
Wang G, Liu Q, Wu J. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of the 17th ACM conference on computer and communications security (CCS). ACM; 2010. pp. 735–7.
Wang G, Liu Q, Wu J. Achieving fine-grained access control for secure data sharing on cloud servers. Concurrency Comput Pract Exp 2011;23(12):1443–64.
Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Public key cryptography (PKC). Springer; 2011. pp. 53–70.
Yu S, Wang C, Ren K, Lou W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: International conference on computer communications (INFOCOM). IEEE; 2010a. pp. 1–9.
Yu S, Wang C, Ren K, Lou W. Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium on information, computer and communications security (ASIACCS). ACM; 2010b. pp. 261–70.

Xin Dong is a Ph.D. candidate in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include networking, information security and privacy, mobile computing and cloud computing. He received his Bachelor degree in computer science and engineering from South China University of Technology (SCUT), Guangzhou, China, in 2010.

Jiadi Yu is an assistant professor in Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. He obtained the PhD degree in Computer Science from Shanghai Jiao Tong University, Shanghai, China, in 2007 and the MS degree in computer science from Xi'an Technological University, Xi'an, China, in 2003. In the past, he has worked as a postdoc at Stevens Institute of Technology, USA, from 2009 to 2011. His research interests include networking, mobile computing, cloud computing and wireless sensor networks.

Yuan Luo received the B.S., M.S., and Ph.D. degrees in applied mathematics from Nankai University, Tianjin, China, in 1993, 1996, and 1999, respectively. From July 1999 to April 2001, he held a postdoctoral position at the Institute of Systems Science, Chinese Academy of Sciences, Beijing, China. From May 2001 to April 2003, he held a postdoctoral position at the Institute for Experimental Mathematics, University of Duisburg-Essen, Essen, Germany. Since June 2003, he has been with the Computer Science and Engineering Department, Shanghai Jiao Tong University, Shanghai, China. His current research interests include coding theory and information theory.

Yingying Chen received the PhD degree in computer science from Rutgers University. She is working as an assistant professor in the Department of Electrical and Computer Engineering at Stevens Institute of Technology. Her research interests include cyber security and privacy, wireless embedded systems, wireless and sensor networks, mobile social networks, and pervasive computing. She was the recipient of the US National Science Foundation CAREER award in 2010. She was the recipient of the Google Research Award in 2010 and the Best Paper Award from the ACM International Conference on Mobile Computing and Networking (MobiCom) in 2011.

Guangtao Xue received his Ph.D. in Computer Science from Shanghai Jiao Tong University in 2004. He is an associate professor in the Department of Computer Science and Engineering at the Shanghai Jiao Tong University. His research interests include mobile networks, social networks, sensor networks, vehicular networks and distributed computing. He is a member of the IEEE Computer Society and the Communication Society.

Minglu Li received his PHD in Computer Software from Shanghai Jiao Tong University in 1996. He is a Full Professor at the Department of Computer Science and Engineering of Shanghai Jiao Tong University. Currently, his research interests include grid computing, services computing, and cloud computing. He has published over 100 papers in important academic journals and international conferences.