A Survey of Attribute-based Access Control with User Revocation in Cloud Data Storage

Chi-Wei Liu[1], Wei-Fu Hsien[2], Chou-Chen Yang[2], and Min-Shiang Hwang[1,3]
(Corresponding author: Min-Shiang Hwang)

Department of Computer Science and Information Engineering, Asia University[1]
No. 500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan (R.O.C.)
(Email: mshwang@asia.edu.tw)
Department of Management Information System, National Chung Hsing University[2]
Department of Medical Research, China Medical University Hospital, China Medical University[3]
No. 91, Hsueh-Shih Road, Taichung 40402, Taiwan (R.O.C.)
(Received June 25, 2015; revised and accepted Aug. 27 & Sept. 23, 2015)

Abstract

Cloud storage service is one of cloud services where cloud service provider can provide storage space to customers. Because cloud storage service has many advantages which include convenience, high computation and capacity, it attracts the user to outsource data in the cloud. However, the user outsources data directly in cloud storage service that is unsafe when outsourcing data is sensitive for the user. Therefore, ciphertext-policy attribute-based encryption is a promising cryptographic solution in cloud environment, which can be drawn up for access control by the data owner to define access policy. Unfortunately, an outsourced architecture applied with the attribute-based encryption introduces many challenges in which one of the challenges is revocation. The issue is a threat to data security in the data owner. In this paper, we survey related studies in cloud data storage with revocation and define their requirements. Then we explain and analyze four representative approaches. Finally, we provide some topics for future research

Keywords: Access control, ciphertext-policy attribute-based encryption, cloud data storage, user revocation

1 Introduction

Cloud computing is a computing technology, and the internet has grown in recent years. It can share the software and hardware resource, and provides resources to a user's computer or mobile device. The user can obtain a more efficient service because cloud computing can integrate resources. Thus, cloud service providers have joined to build cloud environments and provide services to the user. Cloud service providers offer three services including Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The cost for users to rent cloud service is cheaper than the cost for users to build cloud environment [1].

Cloud storage service is the most common and popular service among many cloud services (e.g. Google Drive, Dropbox, Amazon S3 and Microsoft OneDrive) for general users. The user can pay to the cloud server provider based on the amount of usage. Then because cloud storage service provides to access cloud services from web service or applications that utilize the application programming interface (API) by mobile devices (e.g. laptop, table computer, and smart phones), it is convenient to use by users so to achieve ubiquitous service.

Although cloud storage service has many advantages, it also brings a lot of challenging issues which include efficacy and security [12, 17, 36, 44]. One of the serious challenges is protecting the confidentiality of the data. Because the traditional method means the user directly stores the data where data is not encrypted in the cloud storage server, the cloud storage server can understand the upload data of the user. Therefore, if these data are sensitive to users, this is unsafe. In order to ensure that it is safe for the user to upload data to the cloud storage server, a user utilizes an encryption method for processing sensitive data before the user uploads data to the cloud storage server.

For example, a user can utilize a symmetric-key algorithm to encrypt data before uploading to the cloud storage server. When the user needs data, he/she can download data and decrypt it by using a symmetric key. However, it is not suitable when the user shares data with the other users. Because the data owner needs to share their symmetric key with the shared user, the shared user can obtain data on the data owner's permission which contains all the data owner's data in the cloud storage server. Thus, this method is not secure in the situation.

A user chooses the other asymmetric-key algorithm

while the data owner uses a public key to encrypt data before uploading it to the cloud storage server. When the data owner needs data, he/she can download data and decrypt it by using a private key. In the shared situation, the data owner first downloads the shared data and decrypts it. Then, the data owner uses the shared user's public key to encrypt the shared data before uploading to the cloud storage server. However, this method has three problems: first the data owner needs to obtain the correct user's public key where the data owner can encrypt data. Second the same data stored will be repeated in the cloud storage server. Because the data owner will share the encrypted data to each user by their own public key, it will repeat the same data stored in the cloud storage server. Third, when the data owner shares data with a lot users, the data owner takes a lot of resources of the computation in the download and re-encrypted data. To solve these problems of how to design a method that can be able to get the correct user's public key, it only need to store one copy of the shared data in the cloud storage server, which reduces the data owner's resource of the computation.

In order to improve these problems, Sahai and Waters [33] proposed an attribute-based encryption (ABE) scheme where the scheme utilized a user's identity as attribute, and a set of attributes are used to encrypt and decrypt data. Their ABE scheme can resolve these problems including utilizing the attribute of a user's identity to make sure the user's public key, utilizing the ABE to reduce the duplication of data in the cloud storage service and the data owner only need to modify access policy where the data owner can reduce computing resources including downloading, decrypting, re-encrypting and reuploading the entire data.

In the access policy, the ABE has two categories: the key-policy attribute-based encryption (KP-ABE) [9] and the ciphertext-policy attributed encryption (CP-ABE) [4]. The KP-ABE scheme implies that the access policy is attached to the user's private key and use the user's set of attributes to describe the encrypted data. If a set of attributes of the privacy key satisfy the access policy, the user will decrypt the encrypted data. Otherwise, the user cannot obtain the encrypted data. The CP-ABE scheme implies that the access policy is associated with the encrypted data, and use the user's set of attributes to describe the user's private key. If a set of attributes of the encrypted data satisfies the access policy, the user will decrypt the encrypted data. Otherwise, the user cannot obtain the encrypted data.

Nowadays, the outsourced data needs flexible access control for users. The traditional method of access control is a trusted cloud server responsible for the definition and implementation of access control policies. However, users want to be able to share sensitive data and define access policies and the implementation of his/her data with a group of people of their choice. Therefore, it is a desirable method that the access policy of the data will be defined by the data owner.

CP-ABE provides a scalable method of encrypting data where the encrypted user defines the attribute set, and then the decrypted user needs to hold the attribute set to decrypt the ciphertext [4]. Therefore, different users are allowed to decrypt different data block in the different access policies. This effectively reduces depending on the cloud storage server for preventing unauthorized data access.

There are many extended ABE related researches including multi-authority, accountability, proxy re-encryption and revocation. In each ABE scheme, the user needs to get a secret key from the trusted authority which can prove his/her identity, and use the secret key to decrypt data. However, because the authority can decrypt all ciphertexts in a single-authority ABE scheme, the user utilizes a single-authority ABE scheme, which is not proper in the situation there are different departments. Therefore, Chase [7] proposed the first multi-authority ABE scheme which extends a single-authority ABE scheme. Then, the multi-authority ABE schemes were proposed in [6, 7, 8, 11, 18, 26, 27]. In order to achieve secure access control, the ABE scheme needs to prevent accountable key abuse which includes an illegal key sharing among colluding users and misbehavior of the semi-trusted authority containing illegal key distribution or re-distribution. Accountable ABE can be divided in two categories including accountable KP-ABE scheme [37, 42] and accountable CP-ABE scheme [20, 21, 22]. In order to make sharing more efficient, the proxy re-encryption (PRE) is proposed because the user can delegate other to re-encrypt data. However, when the user is not online, the ABE scheme cannot directly use the capability of decryption to others. Therefore, the attribute-based PRE (ABPRE) scheme is proposed [10, 23, 24, 28, 34] which combines the proxy re-encryption with the ABE. A user is able to delegate designated users to decrypt the re-encrypted ciphertext by the associated attributes of designated users.

There are mainly two ways to realize revocation: one is the indirect revocation method [4, 5, 14, 15, 31, 38, 43], and the other is the direct revocation method [2, 25, 30].

Indirect revocation method means the data owner delegates authority to execute revocation which releases a key update material periodically in such a way that only non-revoked users can update their keys. An advantage of the indirect revocation method is that the data owner does not need to know the revocation list. However, the disadvantage of the indirect revocation method is that it requires communication from the authority to all non-revoked users at all time slots in the key update phase. Some related attribute revocable ABE schemes [4, 5, 14, 15, 31, 38, 43] which used the indirect method have been proposed. The direct revocation method means the data owner executes direct revocation which specifies the revocation list while encrypting the ciphertext. An advantage of the direct revocation method over the indirect revocation one is that it does not include the key update phase for all non-revoked users interact-

ing with the authority. However, the disadvantage of the direct revocation method is that it needs the data owner to manage the current revocation list because it is a troublesome problem. Some related attribute revocable ABE schemes [2, 25, 30] which used the direct method have been proposed. Attrapadug and Imai [3] first proposed a hybrid ABE (HR-ABE) scheme which utilized the advantage of both indirect and direct methods. Their scheme allows the data owner to select the encrypted scheme including indirect or direct method. Then, their scheme supports user revocation, but it is unable to achieve attribute revocation. However, it increases the user's secret key in length.

Although there are many ABE related studies, we will focus on user revocation mechanism in the cloud data storage. However, this introduces a number of challenges which utilized ABE to solve the outsourced data. One of the challenge is the revocation of attribute and user. The revocation issue is more difficult in ABE system because each attribute is shared by multiple users. When the revocation of any attribute and single user is in an attribute group, it would affect the other users in the group. It will generate a bottleneck for the rekeying procedure and secure threat in ABE system. Therefore, in this paper, we will survey the problem in attribute-based data access control using CP-ABE for a data outsourcing system.

1.1 Requirement

According to these studies, they provide the basic requirements of function and performance. In our paper, we classify and describe these requirements. Then we use these requirements to analyze the existing scheme in Section 4.

Functional evaluation

1) Data confidentiality: The data owner encrypts the data before uploading data to the cloud. Therefore, the unauthorized user and cloud storage server cannot know the encryption data.
2) Fine-grained access control: Each user respectively has own access right which may be different for each user. Even if the users exist in the same group, their access right may not be the same.
3) Scalability: When the authorized users increase, the cloud storage server can execute efficiently. Therefore, the number of authorized users cannot affect the performance of the cloud storage server.
4) User revocation: If the user leaves the group, the scheme can revoke the user's access right from the cloud storage server. The revoked user cannot access any shared data in the group, because the user does not have access right.
5) Collusion resistant: The revoked user cannot collude with the cloud storage server to obtain the encrypted data which the data owner before sharing the data with the revoked user.
6) Forward secrecy: In an attribute which satisfies the access policy, any user drops the attribute which can be prevented from accessing the plaintext of the subsequent data exchanged after the user drops the attribute.
7) Backward secrecy: In an attribute which satisfies the access policy, any user holds the attribute which can be prevented from accessing the plaintext of the previous data exchanged before he holds the attribute.

Performance evaluation

1) Computing cost: In order to achieve an efficient public auditing, we will analyze the client, TPA and cloud storage service cost on the computing resources.
2) Storage cost: Because the client will upload data to the cloud storage service without the local copy of data files, we will analyze the client, TPA and cloud storage service cost on the storage spaces.

1.2 Our Contribution

Our contribution can be summarized as the following three aspects: First, we survey the previous researches of attribute-based access control with user revocation in the cloud. Then our paper collects and explains basic requirements in the mechanism. Second, we propose four representative approaches and analyze these approaches by our collected requirements. Third, we summarize the conclusion from the analysis and propose research direction in future work.

1.3 Organization

The rest of paper is organized as follows: In Section 2, we review the related work of revocation. We discuss the representative approaches of user revocation in detail in Section 3. In Section 4, we analyze the basic requirement in the representative approaches. Finally, we summarize and discuss the future work in Section 5.

2 Related Work

Recently, some attribute revocable ABE schemes have been proposed [4, 5, 31]. Piretti et al. [31] proposed the time rekeying mechanism where each attribute is associated with an expiration time. Bethencourt et al. [4] improved this solution which utilized the user secret key with a single expiration time. The users need to update their keys frequently, and the authority has lower resource of computation. Boldyreva et al. [5] proposed an efficient revocation scheme for IBE, which utilized binary tree to

build data structure. Their scheme did not use CP-ABE scheme. These schemes were named a coarse-grained revocation because these scheme are not able to immediately rekey on any member change. Furthermore, these schemes have two main problems on scalability and security degradation which include forward and backward secrecy [13, 14, 32, 38]. In the scalability problem, the key authority periodically distributes an update information of key to update the non-revoked users' keys. However, the previous revocation did not consider the scalable distribution where the updated attribute keys distributes the group of users who share the attributes.

In the ABE system, an attribute is supposed to be shared by a group of users. Then it is a considerable situation where the members may change frequently in the group. However, a new user might be able to access the previous encrypted data before the user comes to hold the attributes until the data is re-encrypted with the update attribute keys by periodic rekeying which was named backward secrecy. On the other hand, a revoked user would be able to access the encrypted data until the next expiration time which was named forward secrecy. Therefore, the uncontrolled period has serious vulnerability.

Then many CP-ABE schemes [13, 14, 38, 40, 43] with immediate attribute revocation have been proposed instead of periodic or timed revocation. Yu et al. [43] proposed a scheme which utilized proxy re-encryption with CP-ABE. However, their scheme needed to spend the revocation cost highly where the system public key and users' secret key changed. Hur and Xie et al. [13, 14, 38] proposed efficient attribute revocation schemes which utilized the tree of access policy to encrypt data. The cloud storage server needs to spend high resource of computation because the cloud storage server re-encrypt all the ciphertext with a new generated encrypting key during the attribute revocation. Yang et al. [40] proposed an attribute revocation scheme in CP-ABE where the authority updated the ciphertext and produce new keys that include the new version key, update key, and secret key. However, the authority needed to spend high resource of computation in their scheme.

Yang et al. [41] proposed an improved scheme in CP-ABE which extended multi-authority with attribute revocation. However, the authority needed to enhance efficiency.

User revocation has been observed in the many practical ABE system. Because users may change their attributes frequently, the mechanism of user revocation is essential in many group-based applications [29, 32]. Ibraimi et al. [15] proposed a fine-grained user-level revocation scheme which utilized negative clauses in ABE scheme. When a user is revoked, the user is added to AND of negation of revoked user identities. However, their scheme lacks efficiency of the implementation.

Golle et al. [35] proposed a user revocable scheme which utilized KP-ABE scheme. However, their scheme only supports that the number of attributes associated with a ciphertext is exactly half of the universe size. The previous user-revocable schemes have a drawback on the availability. The availability means the granularity of the user access control between attribute-level or system-level revocation. If a user is revoked from a single attribute group, the user would lose all the access rights on the data sharing system. The previous schemes [25, 30] executed user revocation on system-level, which means the user was revoked from the whole system. However, the system-level revocation scheme is not suitable because the revoked user still has the access right of other data in the system. Therefore, the attribute-level user access control will suit in many practical data outsourcing situation. Attrapadung et al. [2] and Junod et al. [16] proposed user revocation ABE scheme which utilized broadcast encryption scheme on ABE scheme. In Attrapadung et al.'s scheme, the data owner should take full charge of maintaining all the membership lists for each attribute group. However, it is not applicable to the cloud storage architecture because the data owner will no longer be directly in control of data. In the Junod et al.'s scheme, user revocation is achieved by updating the set of identity attributes. However, every user has an identity attribute except for system attribute, which causes the ciphertext growing linearly with the users. Xu et al. [39] proposed a scheme of dynamic user revocation which utilized a delegation key for the cloud storage server to re-encrypt ciphertext. However, the cloud storage server has full control of a revocation list for revoked users. When a user is assigned the revocation list, he/she will lose all access right to the data. Li et al. [19] proposed a revocable identity-based encryption (IBE) scheme in the outsourcing computation. The cloud storage server executes updating secret keys for non-revoked users on the revocation phase. However, two factors that include identities of revoked users and time periods are needed by the secret key generator and the cloud storage server. Zu et al. [45] proposed a new CP-ABE scheme which utilized the access structure of linear secret sharing scheme (LSSS) to define access control in cloud storage service.

3 Representative Approaches

Before introducing representative approaches, we explain the system model in Figure 1, and list all notations (as shown in Table 1) used in this paper.

- The Data Owner: An individual consumer or organization has a lot of data files and needs to store in the cloud. The data owner is responsible for defining (attribute-based) access policy, and encrypting own data under the policy before distributing it.

- Users: A user want to access the data from the data owner. If a user possesses a set of attributes satisfying the access policy of the encrypted data, he/she will be able to decrypt the ciphertext and obtain the data.

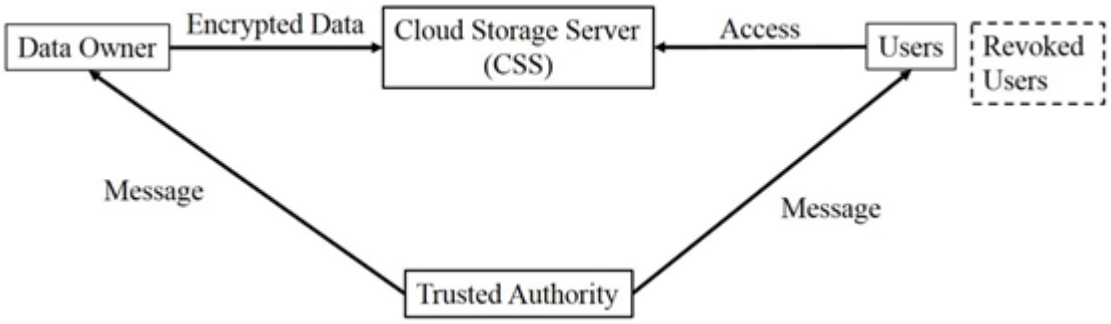

Figure 1: The cloud storage architecture of attribute-based access control

- Cloud Storage Server (CSS): A cloud service provider has huge storage space, computation resource and shared service to provide the clients. It is responsible for controlling the data storage in outside users' access, and provides the corresponding contents.
- Trusted Authority (TA): A trusted organization has expertise and capabilities that the clients do not have. It generates public and secret parameters for ABE, then responsible for issuing, revoking and updated attribute key for the user. It also grants differential access rights to individual users based on their attributes.

3.1 Hur and Noh's Scheme

Hur and Noh [14] was the first to propose an efficient revocation scheme which utilized the access structure of tree to define access control in data outsourcing. Their scheme used CP-ABE scheme to encrypt data and define access policy by the user which was flexible in sharing data with other users. In order to achieve the fine-grained access control, they utilized dual encryption mechanism which took advantage of the ABE and selective group key distribution in each attribute group. They considered the previous user revocable schemes have a limitation which means the granularity of the user access control between attribute-level or system-level revocation. If a user is revoked from a single attribute group in the previous studies [25, 30], the user would lose all the access rights on the system, which means system-level revocation. In the attribute-level revocation, if a user is revoked from a single attribute group, the user would only lose the access right of the attribute group. They proposed the fine-grained revocation to improve coarse-grained revocation because the coarse-grained revocation cannot immediately rekey on any member. The fine-grained revocation can avoid forward and backward secrecy.

Next we will describe their scheme including setup, key generation, data encryption, data re-encryption, data decryption and key update phase.

Setup Phase

The authority chooses two random values α, $\beta \in Z_p^*$, and generates the public parameter $PP = (G_1, g, H_1, h = g^\beta, e(g,g)^\alpha)$ and the master key $MK = (\beta, g^\alpha)$.

Key Generation Phase

The authority uses the master key MK, a set of attributes $\wedge$ and a set of user indices U to generate an attribute key for each user. It chooses a random value $r \in Z_p^*$ which is unique to each user, and a random value $r_j \in Z_p^*$ for each attribute $\lambda_j \in \wedge$ to compute the user u_t's private key $SK_t = (D = g^{(\alpha+r)/\beta}, \forall \lambda_j \in \wedge\text{: } D_j = g^r \cdot H_1(\lambda_j)^{r_j}, D'_j = g^{r_j})$.

The authority generates attribute keys for a set of users U. It uses a set of attributes $\wedge$ and a set of user indices U to generate an attribute keys for each user that identifies with that set $\wedge$. It sends the attribute group AG_j for each attribute $\lambda_j \in \wedge$ to the CSS. For example, if users u_1, u_2, u_3 are connected with $\{\lambda_1, \lambda_2, \lambda_3\}, \{\lambda_2, \lambda_3\}, \{\lambda_1, \lambda_3\}$, respectively. Then the authority sends $AG_1 = \{u_1, u_3\}, AG_2 = \{u_1, u_2\}, AG_3 = \{u_1, u_2, u_3\}$ to the CSS.

The CSS generates key encrypting keys (KEKs) for users in U. It constructs a binary KEK tree for the universe of users μ which will be used to distribute the attribute group keys to users in $U \in \mu$. It assigns each user u_i to the leaf node of the KEK tree, and generates random keys for each leaf node and internal node. Therefore, each user $u_t \in U$ receives the path keys $PathKey_t$ where the path is from the leaf node to the root node.

In the KEK tree (see Figure 2), each node ν_j of the tree holds as KEK, denoted by KEK_j. A set of KEK_j on the path nodes from a leaf to the root are named path keys. For example, the user u_3 stores the path key $PathKey_3 = \{KEK_{10}, KEK_5, KEK_2, KEK_1\}$. Each user u_i is assigned to the leaf node of the KEK tree. Random keys are generated and assigned to each leaf node and internal node.

Each user $u_t \in U$ receives the path keys $PathKey_t$ from its leaf node to the root node of the tree se-

Table 1: Notations

Notation	Significance
G_1, G_2, G_T	A multiplicative cyclic group
e	A bilinear map $e : G_1 \times G_2 \to G_T$
g	A generator of group G_1
p	The prime order of group G_1
q	A much smaller prime than p
H_1	A hash function $H_1 : \{0,1\}^* \to G_1$
H_2	A hash function $H_2 : \{0,1\}^* \to Z_p$
H_3	A hash function $H_3 : G_1 \to Z_p$
M	The message
m_i	A data block of the shared data and will be split into k elements
μ	The universe of users
L	The universe of descriptive attribute
G	The universe of such attribute groups
$\wedge$	A set of attribute
U	A set of user indices
AG	A set of attribute group
K_{λ_i}	The attribute group key

curely. The CSS uses the path keys KEKs to encrypt attribute group keys K_{λ_i} for each AG_i in the re-encryption phase.

Data Encryption Phase

The data owner wants to upload data M to the CSS and sharing data, he/she defines the tree access structure T over the universe of attributes L, and encrypts the data under T.

The data owner chooses a polynomial q_x where x is each node in the access tree T. These polynomials are chosen in a top-down mode which is from the root node R. In the access tree T, the degree d_x of the polynomial q_x be set one less than the threshold value k_x of the node as $d_x = k_x - 1$. Therefore, the root node R is chosen and a random value $s \in Z_p^*$ and set $q_R(0) = s$. Then the root node R sets d_R and other points of the polynomial q_R randomly to define q_R. Any other node x sets $q_x(0) = q_{parent(x)}(index(x))$ and randomly chooses d_x and other points to define q_x.

Then the data owner uses the public parameter PP and the tree of access structure to encrypt the message $M \in G_T$. Therefore, the ciphertext is $CT = (T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H_1(\lambda_y)^{q_y(0)})$ where Y be the set of leaf nodes in the access tree T.

Data Re-Encryption Phase

The CSS uses a set of the membership information for each attribute group $AG \in G$. The attribute group of the access tree is embedded in CT before distributing outsourced data CT. The re-encryption executes user access control from each attribute group on top of the outsourced ciphertext which was encrypted under the attribute-level access control policy by the data owner.

The CSS chooses a random value $K_{\lambda_y} \in Z_p^*$ in the attribute group $AG_y \in AG$ and re-encrypts CT. Therefore, the re-encrypted ciphertext is $CT' = (T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = (H_1(\lambda_y)^{q_y(0)})^{K_{\lambda_y}})$ where Y is the set of leaf nodes in the access tree T. Then the CSS selects the root nodes of the minimum cover sets in the KEK tree which can include all of the leaf nodes connected with users in $AG_i \in AG$. The $KEK(AG_i)$ is constructed from a set of KEKs which include the root nodes of subtrees AG_i. For example, if the attribute groups $AG_i = \{u_1, u_2, u_3, u_4, u_7, u_8\}$, the $KEK(AG_i) = \{KEK_2, KEK_7\}$ because ν_2 and ν_7 are the root node of the minimum cover sets which can cover all of the users in AG_i. If any user $u \notin AG_i$, they would not know any KEK in $KEK(AG_i)$.

Finally, the CSS generates a header message $Hdr = (\forall y \in Y : \{E_K(K_{\lambda_y})\}_{K \in KEK(AG_y)})$ where $E_K(M)$ is a symmetric encryption of a message M under a key K. This encryption is employed for the method to deliver the attribute group keys to valid users. The encryption is $E_K : \{0,1\}^k \to \{0,1\}^k$ a block cipher, where k is the length of the key K. Finally, when the CSS receives the data request from a user, the CSS sends (Hdr, CT') to the user.

Data Decryption Phase

When a user receives the ciphertext (Hdr, CT') from the CSS, he/she first obtains the attribute group keys for all attribute in $\wedge$ that the user holds from Hdr. If a user $u_t \in AG_j$ has a valid attribute λ_j, he/she

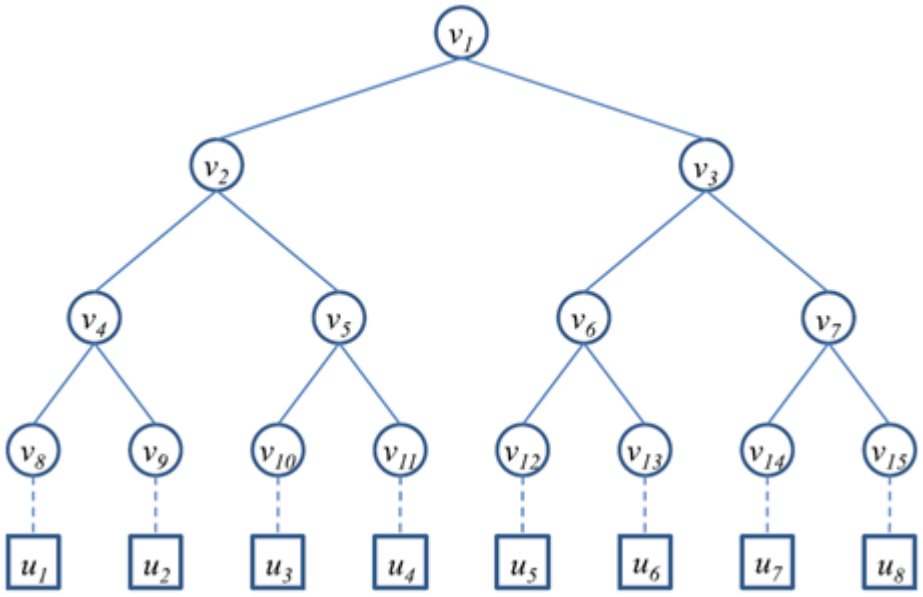

Figure 2: The KEK tree for attribute group key distribution

can decrypt the attribute group key K_{λ_j} from Hdr using a KEK that is common in $KEK(AG_j)$ and $PathKey_t$ where $KEK \in KEK(AG_j) \cap PathKey_t$.

For example, in the attribute groups $AG_i = \{u_1, u_2, u_3, u_4, u_7, u_8\}$, u_3 can decrypt the K_{λ_j} using the path key $KEK_2 \in PathKey_3$. Then the user u_t updates its secret key with the attribute group keys as follows:

$$\begin{aligned} SK_t &= (D, D_j, D'_j) \\ D &= g^{(\alpha+r)/\beta} \\ D_j &= g^r \cdot H_1(\lambda_j)^{r_j} \\ D'_j &= (g^{r_j})^{1/K_{\lambda_j}}, \forall \lambda_j \in \wedge. \end{aligned}$$

Then the user uses a private key SK and K_{λ_x} to decrypt the encrypted ciphertext CT' in the recursive function as $DecryptNode(CT', SK, x)$.

If the node x is a leaf node and $\lambda_x \in \wedge$ and $u_t \in AG_x$, then it computes

$$\begin{aligned} & DecryptNode(CT', SK, x) \\ = & \frac{e(D_x, C_x)}{e(D'_x, C'_x)} \\ = & \frac{e(g^r H(\lambda_x)^{r_x}, g^{q_x(0)})}{e((g^{r_x})^{1/K_{\lambda_x}}, (H(\lambda_x)^{q_x(0)})^{K_{\lambda_x}})} \\ = & e(g,g)^{rq_x(0)}. \end{aligned}$$

If $u_t \notin GA_x$, the user u_t cannot compute the values $e(g,g)^{rq_x(0)}$, as the exponent of D'_x in SK cannot include the inverse of the exponent K_{λ_x} of C'_x. If $\lambda_x \notin \wedge$ or $u_t \notin GA_x$, the $DecryptNode(CT', SK, x)$ will output invalid.

If the node x is a non-leaf node, the $DecryptNode(CT', SK, x)$ can be named from all nodes z which are children of x. For all nodes z call $DecryptNode(CT', SK, z)$ which use Lagrange coefficient to compute and obtain $e(g,g)^{rq_x(0)}$.

Therefore, if the access tree T is satisfied by $\wedge$, the user has valid memberships for each attribute group AG_i for all $\lambda_i \in \wedge$. Let $A = DecryptNode(CT', SK, R) = e(g,g)^{rs}$.

Finally, the user decrypts the ciphertext.

$$\begin{aligned} \frac{\tilde{C}}{e(C,D)/A} &= \frac{Me(g,g)^{\alpha s}}{e(h^s, g^{(\alpha+r)/\beta})/e(g,g)^{rs}} \\ &= M. \end{aligned}$$

Key Update Phase

When a user comes to hold or drop an attribute, the corresponding key should be updated to avoid backward and forward secrecy on the previous or subsequent encrypted data. The key update procedure is executed by the authority when the user requests to join or leave on the attribute group. The authority receives the request, and sends the updated membership list of the attribute group to the CSS. Then the CSS receives the update request, and computes the corresponding attribute group key.

The CSS selects a random value $s' \in Z_p^*$ and a random value K'_{λ_i} which is different from the previous attribute group key $K_{\lambda_i} \neq K'_{\lambda_i}$. Then it re-encrypts the ciphertext CT using the public parameters PP

and public key PK as

$$\begin{aligned} CT' &= (T, \tilde{C}, C, C_i, C_i') \\ \tilde{C} &= Me(g,g)^{\alpha(s+s')} \\ C &= h^{s+s'} \\ C_i &= g^{q_i(0)+s'} \\ C_i' &= (H_1(\lambda_i)^{q_i(0)+s'})^{K'_{\lambda_i}} \end{aligned}$$

$\forall y \in Y : C_y = g^{q_i(0)+s'}, C_y' = (H_1(\lambda_y)^{q_y(0)+s'})^{K_{\lambda_y}})$.

In the other attribute groups, the attribute group keys do not necessarily need to be uploaded because they will not be affected by the membership changes.

The CSS chooses a new minimum set to cover the original attribute group AG_i, and the new set includes a new joining user who comes to hold an attribute λ_i (or exclude a leaving user who come to drop an attribute λ_i).

The CSS generates a new header message with the updated $KEK(AG_i)$ as

$$\begin{aligned} Hdr = (&\{E_K(K'_{\lambda_i})\}_{K \in KEK(AG_i)}, \\ &\{E_K(K_{\lambda_y})\}_{K \in KEK(AG_i)}, \forall y \in Y). \end{aligned}$$

Finally, the CSS responds new header message and the ciphertext.

3.2 Hur's Scheme

Hur [13] proposed an improved security scheme which considered a key escrow problem and user revocation in attribute-based data sharing. Their scheme used CP-ABE scheme to encrypt data and define access policy by the user which was flexible in sharing data with other users. Because the authority generates users' private keys by using the authority's the master key to users' associated set of attributes in the attribute-based encryption, the authority can decrypt every ciphertext addressed to specific users. This problem could generate a potential threat in the data sharing system of data confidentiality or privacy. Therefore, they designed the scheme where the authority and the CSS generated the user's secret key together that could avoid the key escrow problem. Then they considered the key revocation where the user may change their associate attributes. Therefore, the key revocation or update for each attribute is necessary to make system secure.

Next we will describe their scheme including setup, key generation, data encryption, data re-encryption, data decryption and key update phase.

Setup Phase

The authority chooses a random value $\beta \in Z_p^*$ and computes $h = g^\beta$. The public parameter $PP = (G_1, g, H_1, H_3)$, the public key $PK_A = h$ and the master key $MK_A = \beta$.

The CSS chooses a random value $\alpha \in Z_p^*$. The public key $PK_C = e(g,g)^\alpha$ and the master key $MK_C = g^\alpha$.

The CSS chooses another random value $\Upsilon \in Z_p^*$, and generates another public key $PK_C^{agree} = g^\Upsilon$ while keeping Υ as a secret.

Key Generation Phase

The authority needs to authenticate a user u_t which exists in U. If the result is true, the authority chooses a random value $r_t \in Z_p^*$ which is a unique secret for the user. Then the authority and the CSS construct a secure 2PC protocol, which combine the values (r_t, β) from the authority with the value α from the CSS. Therefore, the secure 2PC protocol is the value $x = (\alpha + r_t)\beta$.

1) The CSS chooses a random value $\tau \in Z_p^*$, computes $A = g^{\frac{x}{\tau}} = g^{\frac{(\alpha+r_t)\beta}{\tau}}$, and then sends it to the authority.
2) The authority computes $B = A^{1/\beta^2} = g^{\frac{\alpha+r_t}{\tau\beta}}$, and sends it to the CSS.
3) The CSS generates a personalized key component $D = B^\tau = g^{\frac{\alpha+r_t}{\beta}}$.
4) The authority uses a set of attributes $\wedge$ that a user u_t is entitled to have, and generates a set of attribute keys identified with that set and the secret value r_t. The authority chooses a random value $r_j \in Z_p^*$ for each attribute $\lambda \in \wedge$. Then it computes a user u_t's the attribute keys $SK_{A,u_t} = (\lambda_j \in \wedge : D_j = g^{r_t} H_1(\lambda_j)^{r_j}, D'_j = g^{r_j})$ to the CSS.
5) The CSS's personalized key component SK_{u_t} for a user u_t as $SK_{C,u_t} = D = g^{(\alpha+r_t)/\beta}$. Then the user u_t can obtain its whole secret key

$$\begin{aligned} SK_{u_t} &= (SK_{C,u_t}, SK_{A,u_t}) \\ &= (D, D_j, D'_j). \\ D &= g^{(\alpha+r_t)/\beta} \\ D_j &= g^{r_t} \cdot H(\lambda_j)^{r_j}, \forall \lambda_j \in \wedge \\ D'_j &= g^{r_j}. \end{aligned}$$

The CSS also generates another encrypting key (KEK) $SK_{u_t}^{agree} = H(ID_t)^\Upsilon = Q_t^\Upsilon$ for the user, which will be used for selective attribute group key distribution.

Data Encryption Phase

The data owner wants to upload data M to the CSS and sharing data, he/she defines the tree access structure T over the universe of attributes L, and encrypts the data under T.

The data owner chooses a polynomial q_x where x is each node in the access tree T. These polynomials are chosen in a top-down method which is from the root node R. In the access tree T, the degree d_x of the polynomial q_x is set one less than the threshold value k_x of the node as $d_x = k_x - 1$. Therefore, the root node R chooses a random value $s \in Z_p^*$ and

set $q_R(0) = s$. Then the root node R sets d_R other points of the polynomial q_R randomly to define q_R. Any other node x sets $q_x(0) = q_{parent}(x)(index(x))$ and randomly chooses d_x other points to define q_x.

The data owner uses the public parameter and the tree of access structure to encrypt the message $M \in G_T$. Therefore, the ciphertext is $CT = (T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H_1(\lambda_y)^{q_y(0)})$ where Y is the set of leaf nodes in the access tree T.

Data Re-encryption Phase

The CSS uses a set of the membership information for each attribute group $AG \subseteq Q$. The attribute group of the access tree is embedded in CT before distributing outsourced data CT. The re-encryption executes user access control from each attribute group on top of the outsourced ciphertext which was encrypted under the attribute-level access control policy by the data owner.

The CSS chooses a random value $K_{\lambda_y} \in Z_p^*$ in the attribute group $GA_y \in GA$ and re-encrypts CT. Therefore, the re-encrypted ciphertext is $CT' = (T, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \forall y \in Y : C_y = g^{q_y(0)}, C'_y = (H_1(\lambda_y)^{q_y(0)})^{K_{\lambda_y}})$ where Y is the set of leaf nodes in the access tree T. Then, it selects $\rho, R \in Z_p^*$, and for all $u_t \in AG$ computes $x_t = H_3(e(Q_t^\rho, PK_C^{agree}))$. For all $AG_y \subset AG$ constructs the polynomial function $f^y(x) = \prod_{i=1}^{m}(x - x_i) = \sum_{i=0}^{m} a_i x^i (\mathrm{mod} p)$, where $AG_y = \{u_1, u_2, \cdots, u_m\}$ and the exponential function $\{P_0, \cdots, P_m\} \equiv \{g^{a_0}, \cdots, g^{a_m}\}$, where m is the number of users in the attribute group. It constructs $Hdr_y = \{K_{\lambda_y} \cdot P_0^R, P_1^R, \cdots, P_m^R\}$ and generates a header message $Hdr = (g^\rho, \cdots, \forall y \in Y : Hdr_y)$. Finally, when the CSS receives the data request from a user, the CSS sends (Hdr, CT') to the user.

Data Decryption Phase

A user receives the ciphertext (Hdr, CT') from the CSS, he/she first obtains the attribute group keys for all attributes in $\wedge$ that the user holds from Hdr. If a user $u_t \in AG_j$ has a valid attribute λ_j, he/she can decrypt the attribute group key K_{λ_j} from Hdr. The user u_t uses the KEK $SK_{u_t}^{agree}$ and g^ρ and computes $x_t = H_1(e(g^\rho, SK_{u_t}^{agree}))$. Then, the user u_t computes $K_{\lambda_j} \cdot P_0^R \cdot \prod_{i=1}^{m}(P_i^R)^{x_t^i} = K_{\lambda_j} \cdot g^{Rf^j(x_t)} = K_{\lambda_j}$, where $m = |AG_j|$. The user u_t updates its private key with the attribute group keys $SK_{u_t} = (D = g^{(\alpha + r_t)/\beta}, \forall \lambda_j \in \wedge : D_j = g^{r_t} \cdot H(\lambda_j)^{r_j}, D'_j = (g^{r_j})^{(1/K_{\lambda_j})})$.

Then, the user uses private key SK and K_{λ_x} to decrypt the encrypted ciphertext CT' in the recursive function as $DecryptNode(CT', SK, x)$.

If the node x is a leaf node and $\lambda_x \in \wedge$ and $u_t \in AG_x$, then it computes

$$\begin{aligned} & DecryptNode(CT', SK, x) \\ = \ & \frac{e(D_x, C_x)}{e(D'_x, C'_x)} \\ = \ & \frac{e(g^r \cdot H(\lambda_x)^{r_x}, g^{q_x(0)})}{e((g^{r_x})^{\frac{1}{K_{\lambda_x}}}, (H(\lambda_x)^{q_x(0)})^{K_{\lambda_x}})} \\ = \ & e(g,g)^{rq_x(0)}. \end{aligned}$$

If $u_t \notin GA_x$, the user u_t cannot compute the values $e(g,g)^{rq_x(0)}$, as the exponent of D'_x in SK cannot include the inverse of the exponent K_{λ_x} of C'_x.

If $\lambda_x \notin \wedge$ or $u_t \notin GA_x$, the $DecryptNode(CT', SK, x)$ will output invalid value.

If the node x is a non-leaf node, the $DecryptNode(CT', SK, x)$ can be named from all nodes z which are children of x. For all nodes z call $DecryptNode(CT', SK, z)$ which use Lagrange coefficient to compute and obtain $e(g,g)^{rq_x(0)}$. Therefore, if the access tree T is satisfied by $\wedge$, and the user has valid memberships for each attribute group AG_i for all $\lambda_i \in \wedge$. Let $A = DecryptNode(CT', SK, R) = e(g,g)^{r_t s}$.

The user decrypts the ciphertext

$$\begin{aligned} \frac{\tilde{C}}{(e(C,D)/e(g,g)^{r_t s})} &= \frac{\tilde{C}}{(e(h^s, g^{(\alpha + r_t)/\beta})/e(g,g)^{r_t s})} \\ &= \frac{\tilde{C}}{(e(g^{\beta s}, g^{(\alpha + r_t)/\beta})/e(g,g)^{r_t s})} \\ &= \frac{Me(g,g)^{\alpha s}}{e(g,g)^{s\alpha}} \\ &= M. \end{aligned}$$

Key Update Phase

When a user comes to hold or drop an attribute, the corresponding key should be updated to avoid backward and forward secrecy on the previous or subsequent encrypted data. The key update procedure is executed by the authority when the user requests to join or leave on the attribute group. The authority receives the request, and sends the updated membership list of the attribute group to the CSS. Then the CSS receives the update request, and computes the corresponding attribute group key.

The CSS selects a random value $s' \in Z_p^*$ and a random value K'_{λ_i} which is different from the previous attribute group key $K_{\lambda_i} \neq K'_{\lambda_i}$. Then the CSS re-encrypts the ciphertext CT using the public param-

eters PP and public key PK as

$$\begin{aligned} CT' &= (T, \tilde{C}, C, C_i, C_i', \forall y \in Y\{i\} : C_y, C_y'). \\ \tilde{C} &= Me(g,g)^{\alpha(s+s')} \\ C &= h^{s+s'} \\ C_i &= g^{q_i(0)+s'} \\ C_i' &= (H_1(\lambda_i)^{q_i(0)+s'})^{K'_{\lambda_i}} \\ C_y &= g^{q_i(0)+s'} \\ C_y' &= (H_1(\lambda_y)^{q_y(0)+s'})^{K_{\lambda_y}}. \end{aligned}$$

In the other attribute groups, the attribute group keys do not necessarily need to be uploaded because they will not be affected by the membership changes.

The CSS generates a new polynomial function $f^i(x)$ with a new attribute group AG_i including a new joining user who comes to hold an attribute λ_i (or excluding a leaving user who comes to drop an attribute λ_i). The CSS generates a new header message Hdr_i with the attribute group key K'_{λ_i} as $Hdr = (g^\rho, Hdr_i, \forall y \in Y\{i\} : Hdr_y)$, where the header message Hdr_y are the same before. Finally, the CSS responds new header message and the ciphertext.

3.3 Yang et al.'s Scheme

Yang et al. [40] proposed an attribute revocation scheme in CP-ABE which utilized the access structure of linear secret sharing scheme (LSSS) to define access control in cloud storage service. Their scheme did not need to re-encrypt the ciphertext by the CSS, because they considered to be unsafe from the semi-trusted CSS re-encrypting. However, the authority needed to spend high resource of computation in their scheme. Because their key update had three parts including update key generation, secret key update and ciphertext update, the authority needed to update the ciphertext and produce new keys that include the new version key, update key, and secret key, the ciphertext and producing new keys in their scheme

Next we will describe their scheme including setup, key generation, data encryption, data decryption and key update phase.

Setup Phase

The authority chooses random values $\alpha, \beta, \Upsilon, a \in Z_p$, and generates the public parameter $PP = \{g, g^a, g^{1/\beta}, g^\beta, e(g,g)^\alpha\}$, and the master keys are $MK = \{\alpha, \beta, \Upsilon, a\}$.

For each attribute x, the authority generates a random value $v_x \in Z_p$ as the attribute version number $VK_x = v_x$. Then the authority utilizes VK_x to generate a public attribute key $PK_x = (PK_{1,x} = H_1(x)^{v_x}, PK_{2,x} = H_1(x)^{v_x \Upsilon})$.

Key Generation Phase

When a user joins the system, the authority first assigns a set of attribute S to this user according to its identity.

The authority uses the master key MK, a set of attributes S that describes the secret key, and the corresponding set of attributes the user's secret key $SK = (K = g^{\alpha/\beta} \cdot g^{(at)/\beta}, L = g^t, \forall x \in S : K_x = g^{t\beta^2} \cdot H_1(x)^{v_x t\beta})$. Finally, the authority sends SK to the user in a secure channel.

Data Encryption Phase

The data owner first divides the data $M = \{m_1, m_2, \cdots, m_n\}$ according to the logic granularity. Then it uses symmetric encryption methods to encrypt the data as the content key $k = \{k_1, k_2, \cdots, k_n\}$ where $k_i = E_K(m_i)$.

The data owner uses the public parameter PP, a set of public attribute key $\{PK_x\}$, a content key k and a LSSS access structure (TM, ρ). Let TM be a $l \times n$ matrix, where l means the number of attributes involved in the encryption. The function ρ which is associated rows of TM to attributes is a limited injective function. It first chooses a random encryption exponent $s \in Z_p$ and a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in Z_p^n$, where $y_2, \cdots, y_n$ are used to share the encryption exponent s. For $i = 1$ to l, it computes $\lambda_i = \vec{v} \cdot TM_i$, where TM_i is the vector corresponding to the ith row of TM. Then it chooses random values $r_1, r_2, \cdots, r_l \in Z_p$ and computes the ciphertext $CT = (C = ke(g,g)^{\alpha s}, C' = g^{\beta s}, \forall i = 1, \cdots, l, C_i = g^{a\lambda_i}(g^\beta)^{-r_i v_\rho(i)}, D_{1,i} = H_1(\rho(i))^{v_{\rho(i)} r_i \Upsilon}, D_{2,i} = g^{r_i/\beta})$.

Data Decryption Phase

The user receives the data from the CSS. Only the attribute that the user possesses satisfies the access structure defined in the ciphertext CT, so the user can get the data component successfully. Users with different attributes will be able to decrypt different number of data components, such that they can get different granularities of information from the same data.

The user uses a ciphertext CT attached with the access structure (TM, ρ) and the secret key for a set of attribute S. The user's attribute set S satisfies the access structure and let $I \subset \{1, 2, \cdots, l\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then it chooses a set of constants $\{w_i \in Z_p\}_{i \in I}$ and reconstructs the encryption exponent as $s = \sum_{i \in I} w_i \lambda_i$ if $\{\lambda_i\}$ are valid shares of the secret s according to TM. Then the user first

computes

$$\begin{aligned}
& \frac{e(C', K)}{\prod_{i\in I}(e(C_i, L)e(D_{2,i}, K_{\rho(i)}))^{w_i}} \\
= & \frac{e(g^{\beta s}, g^{\alpha/\beta} \cdot g^{(at)/\beta})}{\prod_{i\in I}(e(g^{a\lambda_i} H_1(\rho(i))^{-v_{\rho(i)} r_i}, g^t) \cdot e(g^{r_i/\beta}, H_1(\rho(i))^{v_{\rho(i)} t\beta}))^{w_i}} \\
= & \frac{e(g,g)^{\alpha s} e(g,g)^{sat}}{e(g,g)^{at\sum_{i\in I}\lambda_i w_i}} \\
= & e(g,g)^{\alpha s}.
\end{aligned}$$

It can then decrypt the content key $k = C/e(g,g)^{\alpha s}$. The user uses a symmetric key and the content keys to further decrypt the data $D_K(k) = m$.

Key Update Phase

1) Update Key Generation by the Authority: When these is an attribute revocation, the authority generates the update key by using the master key MK and the current version key $VK_{x'}$ of the revoked attribute x'.

   It chooses a random value $v'_{x'} \in Z_p$ where $v'_{x'} \neq v_{x'}$ and generates a new attribute version key $VK'_{x'}$.

   The authority uses $VK'_{x'}$ to computes the update key $UK_{x'} = (UK_{1,x'} = \frac{v'_{x'}}{v_{x'}}, UK_{2,x'} = \frac{v_{x'} - v'_{x'}}{v_{x'}\Upsilon}$. The authority sends the update key $UK_{x'}$ to the CSS. Then the authority also updates the public attribute key of the revoked attribute x' as

$$\begin{aligned}
PK'_{x'} &= (PK'_{1,x'}, PK_{2,x'}). \\
PK'_{1,x'} &= (PK_{1,x'})^{UK_{1,x'}} = H_1(x')^{v'_{x'}} \\
PK_{2,x'} &= (PK_{2,x'})^{UK_{1,x'}} = H_1(x')^{v'_{x'}\Upsilon}.
\end{aligned}$$

   Finally, the authority broadcasts a message to all the users that the public attribute key of the revoked attribute x' is updated.

2) Secret Key Update by Non-revoked Users: Each non-revoked user sends two components $L = g^t$ and $K_{x'}$ of the secret key SK to the authority. The authority receives these components and computes a new components $K'_{x'} = (K_{x'}/L^{\beta^2})^{UK_{1,x'}} \cdot L^{\beta^2} = g^{t\beta^2} \cdot H_1(x')^{v'_{x'} t\beta}$. Then it returns the new component $K'_{x'}$ to the non-revoked user.

   The non-revoked user's secret key is updated by replacing the component $K_{x'}$ associated with the revoked attribute x' with the new one $K'_{x'}$ as $SK' = (K, L, K_{x'}, \forall x \in S\{x'\} : K_x)$.

3) Ciphertext Update by Cloud Server: The CSS receives the update key UK_x from the authority and updates the ciphertext associated with the revoked attribute x'. The CSS uses the ciphertext CT and the update key $UK_{x'}$ to update the ciphertext $CT' = (\tilde{C} = C, \tilde{C'} = C', \forall i = 1, cdots, l : \tilde{D_{2,i}} = D_(2,i)$, if $\rho(i) \neq x' : \tilde{C_i} = C_i, \tilde{D_{1,i}} = D_{1,i}$, if $\rho(i) = x' : \tilde{C_i} = C_i \cdot (D_{1,i})^{UK_{2,x'}}, \tilde{D_{1,i}} = (D_{1,i})^{UK_{1,x'}})$.

3.4 Zu et al.'s Scheme

Zu et al. [45] proposed a new CP-ABE scheme which utilized the access structure of linear secret sharing scheme (LSSS) to define access control in cloud storage service. Their scheme had efficient revocation and fine-grained access control. Their scheme combined proxy re-encryption with CP-ABE to achieve the user and attribute revocation. In their scheme, the authority generated two secret keys of the user where one sends to the user, and the other sends to the cloud storage server. When the authority removes a user's attribute, their scheme would not affect other users' access privileges with this attribute. Finally, their scheme could reduce the load from the authority on the revocation.

Next we will describe their scheme including setup, key generation, data encryption, data re-encryption, and data decryption phase.

Setup Phase

The authority chooses random values $\alpha_1, \alpha_2, a \in Z_p$ such that $\alpha = \alpha_1 + \alpha_2 \bmod p$, and generates the public parameter $PP = \{G_1, g, H_1, e(g,g)^{\alpha}, g^a\}$, and the master keys are $MK = \{\alpha_1, \alpha_2, g^{\alpha}\}$.

Key Generation Phase

The authority uses one part of the master key α_1, a set of attributes S and chooses a random value $t \in Z_p$. The user's secret key is generated as $SK_1 = \{K = g^{\alpha_1} g^a t, L = g^t, \forall x \in S : K_x = H_1(x)^t\}$. The authority uses the other part of the master key α_2 to generate the delegation key $SK_2 = \{D_c = g^{\alpha_2}\}$ for the CSS.

Data Encryption Phase

When a data owner wants to upload its data M to the CSS for sharing, the data owner uses the public parameters PP and an LSSS access structure (TM, ρ) to encrypt a message M. Let TM be an $l \times n$ matrix, TM_i be the vector corresponding to the ith row of TM. The function ρ which is associated with rows of TM to attributes is a limited injective function.

The data owner chooses random values $r_1, r_2, \cdots, r_l \in Z_p$ and a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in Z_p^n$. These elements of vector $\vec{v}$ will be used to share the encryption exponent s. For $i = 1$ to l, computes $\lambda_i = TM_i \vec{v}$. The ciphertext is published as $CT = \{\tilde{C} = Me(g,g)^{\alpha s}, C = g^s, \forall 1 \leq i \leq l, \rho(i) \in S : C_i = g^{\lambda_i} H_1(\rho(i))^{r_i}, D_i = g^{r_i}\}$ along with a description of (TM, ρ).

Data Re-encryption Phase

When a user comes to hold or drop an attribute,

the corresponding key associated with the attribute should be updated. Because the re-encryption can prevent the user from accessing the previous or subsequent re-encrypted data for backward or forward secrecy, the key associated with the attribute needs to be updated. We denote ID_i as the identity of the user i.

1) If there is no attribute revocation, the CSS uses a random $k \in Z_p$ to encrypt the delegation key g^{α_2} and the ciphertext $CT = (D'_c = (g^{\alpha_2})^k, \tilde{C} = Me(g,g)^{\alpha s}, C = g^s, C' = g^{s/k}, C'_i = g^{a\lambda_i} H_1(\rho(i))^{r_i} H_1(\rho(i))^k, D'_i = g^{r_i} g^k)$.
   The re-encrypted ciphertext $\tilde{CT} = \{CT', D'_c\}$ is then sent to the user, where $CT' = \{\tilde{C}, C, C', \{C'_i, D'_i\}_{i=1,\cdots,l}\}$.
2) If there is an attribute x' revocation from a user ID_j where ID_j means the identity of the user j, the CSS encrypts a random key $\nu_{x'} \in Z_p$ as $\tilde{C}$ under the access structure (TM, ρ) for those users $ID_i, i \neq j$ who hold the revoked attribute but not been revoked. The method of encrypting random keys and decrypting ciphertext $\tilde{C}$ is similar to that of Liang et al. scheme [35].

Then the CSS utilizes a random value $k \in Z_p$ to encrypt the delegation key g^{α_2} and the ciphertext $CT = (D'_c = (g^{\alpha_2})^k, \tilde{C} = Me(g,g)^{\alpha s}, C = g^s, C' = g^{s/k}, \forall i = 1, 2, \cdots, l, C'_i = g^{a\lambda_i} H_1(\rho(i))^{r_i} H_1(\rho(i))^k$, for $\rho(i) \neq x' : D'_i = g^{r_i} g^k$; for $\rho(i) = x : D'_i = (g^{r_i} g^k)^{1/\nu_{(\rho(i))}})$.

The re-encrypted ciphertext $\tilde{CT} = \{CT', D'_c, \tilde{C}\}$ is then sent to the users, where $CT' = \{\tilde{C}, C, C', \{C'_i, D'_i\}_{i=1,\cdots,l}\}$.

Data Decryption Phase

A user receives the ciphertext $\tilde{CT}$ for access structure (TM, ρ), and uses the private key SK_1 for a set of attributes S to decrypt:

1) If there is no attribute revocation, the user computes

$$A = \frac{\prod_{i \in I} e(C'_i, L)^{w_i}}{\prod_{i \in I} e(D'_i, K_{\rho(i)})^{w_i}} = e(g,g)^{ats}.$$

   The user decrypts the ciphertext

$$\frac{\tilde{C} \cdot A}{e(C', D'_c) e(C, K)} = \frac{Me(g,g)^{\alpha s} \cdot e(g,g)^{ats}}{e(g^s, g^{\alpha_2 k} e(g^s, g^{at}))} = M.$$

2) If there is an attribute x' revocation from a user ID_j. The user $ID_i, i \neq j$ holds the revoked attributes S and satisfies with the access structure (TM, ρ), then the user decrypts $\tilde{C}$ using SK_1 and obtains $\nu_{x'}$ to update the secret key $K_{x'}$ as $K_{x'} = (H(x')^t)^{\nu_{x'}}$. Otherwise, he/she cannot get the updated secret key $K_{x'}$.

The first step of decryption of $\tilde{CT}$ proceeds in the following: for $\rho(i) \neq x' : B_i = \frac{e(C'_i, L)^{w_i}}{e(D'_i, K_{\rho(i)})^{w_i}} = e(g,g)^{at\lambda_i w_i}$, for $\rho(i) = x' : B_i = \frac{e(C'_i, L)^{w_i}}{e(D'_i, \tilde{K_{\rho(i)}})^{w_i}} = e(g,g)^{at\lambda_i w_i}$, $A = \prod_{i \in I} B_i = e(g,g)^{ats}$. The user decrypts the ciphertext $\frac{\tilde{C} \cdot A}{e(C', D'_c) e(C, K)} = \frac{Me(g,g)^{\alpha s} \cdot e(g,g)^{ats}}{e(g^s, g^{\alpha_2 k}) e(g^s, g^{at})} = M$.

4 Analysis

In the section, we will analyze these schemes [13, 14, 40, 45] which contain functional requirement, security and performance. And we also use the tables to present a corresponding requirement in each scheme.

4.1 Functional Evaluation

In Table 2, we will analyze several functional requirements: fine-grained access control, scalability, user accountability, user revocation, collusion resistant, forward secrecy and backward secrecy in the representative approaches. Almost schemes can achieve these functional requirement including data confidentiality, fine-grained access control, user revocation, collusion resistant, forward secrecy and backward secrecy. In K. Yang et al.'s scheme, when an attribute is revoked, non-revoked users need to update their secret keys. Therefore, their scheme did not satisfy the scalability.

4.2 Performance Evaluation

We will analyze four phases: setup phase, key generation phase, data encryption phase, data re-encryption phase; data decryption phase and user revocation phase in the four entities include data owner, user (the group user), cloud storage server (CSS) and the authority. Before we analyze the performance evaluation, first we introduce the notations in Table 3.

In Table 4, we analyze four schemes how to execute a setup phase. Hur's scheme needs the CSS to generate the public key, the master key and another key because they considered the key escrow problem. K. Yang et al.'s scheme spent more computing resources.

In Table 5, we analyze four schemes how to execute a key generation phase. Because the CSS executes partly computation, the authority could reduce computation in Hur's scheme. However, Zu et al.'s scheme needed lower computation in these schemes when a set of attributes are smaller.

In Table 6, we analyze four schemes how to execute a data encryption phase. K. Yang et al.'s scheme needed more computing resources, but they did not execute re-encryption phase. Hur and Noh's scheme and Hur's scheme needed less computing resource.

Table 2: Comparison of functional requirements

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
Data confidentiality	Yes	Yes	Yes	Yes
Fine-grained access control	Yes	Yes	Yes	Yes
Scalability	Yes	Yes	No	Yes
User revocation	Yes	Yes	Yes	Yes
Collusion resistant	Yes	Yes	Yes	Yes
Forward secrecy	Yes	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes	Yes

Table 3: Notations

Notation	Significance
T_{sym}	The computing time of symmetric encryptions
T_{Ge}	The computing time of exponentiation in group operation
T_B	The computing time of bilinear pairing
T_{Mul}	The computing time of multiplication
T_{Div}	The computing time of division
T_{Add}	The computing time of addition
T_{Sub}	The computing time of subtraction
T_{GM}	The computing time of multiplication in group operation
T_h	The computing time of hash function
$\wedge$	A set of attributes
i	A set of revoked attributes
m	The number of users in the group

Table 4: Comparison of computation in the setup phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
CSS		$3T_{Ge}$		
Authority	$2T_{Ge}$	$1T_{Ge}$	$5T_{Ge} + 2T_h$	$3T_{Ge} + 1T_A$

Table 5: Comparison of computation in the key generation phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
CSS		$3T_{Ge} + T_h$		
Authority	$2T_{Ge} + T_{GM}$ $+ \wedge (3T_{Ge} + T_{GM} + T_h)$	$T_{Ge} + T_{Mul}$ $+ \wedge (3T_{Ge} + T_{GM} + T_h)$	$4T_{Ge} + T_{GM}$ $+ \wedge (5T_{Ge} + T_{GM} + T_h)$	$4T_{Ge} + T_{GM}$ $+ \wedge (T_{Ge} + T_h)$

Table 6: Comparison of computation in the data encryption phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
Data owner	$2T_{Ge} + T_{GM}$ $+ \wedge (2T_{Ge} + T_h)$	$2T_{Ge} + T_{GM}$ $+ \wedge (2T_{Ge} + T_h)$	$T_{sym} + 2T_{Ge} + T_{GM}$ $+ \wedge (7T_{Ge} + T_{GM} + T_h)$	$2T_{Ge} + T_{GM}$ $+ \wedge (3T_{Ge} + T_{GM} + T_h)$

In Table 7, we analyze four schemes how to execute a data re-encryption phase. Hur and D. Noh's scheme needed less computing resource in these scheme. K. Yang et al. did not execute data re-encryption.

In Table 8, we analyze four schemes how to execute a data decryption phase. Hur's scheme needed to spend more computing resource because they considered details on the decryption. Zu et al.'s scheme was better in these scheme because they need less computing resources.

In Table 9, we analyze four schemes how to execute a key update phase. K. Yang et al.'s scheme needed to compute the CSS and the authority together. Although Zu et al.'s scheme did not support key update, their schemes executed re-encryption in the situation of attribute revocation. Hence, we describes the situation of attribute revocation in key update.

5 Conclusion and Future Work

Conclusion

Although cloud data storage has many advantages, it also bring many challenges. When the cloud service provider provides a semi-trusted cloud server, it may steal clients' data which is serious issues. Therefore, data confidentiality and access control are important issues in cloud storage. Then cloud data can share own data with other in cloud platform. Therefore, the access controls which users to share the data together, and a user leaves the access privilege of the data. Although there are many kinds of access control schemes, they have to apply the restriction of cloud environment. Because users store data in the cloud storage, they cannot control their data. Attribute-based encryption is a promising scheme in data security which can limit the data to access control. Ciphertext-policy attribute-based encryption is an applicable scheme in the cloud data storage which encrypts the data and defines access structure from the user.

Therefore, we survey the previous researches of attribute-based access control with user revocation in the cloud. Then we collect and explain basic requirements in the mechanism. We analyze these approaches by using function and performance evaluation. Finally, in this paper, we provide the future development of CP-ABE with user revocation.

Future Work

For future developments, we will focus on the following areas of particular interest. Efficiency: The authority needs to generate every user' key and other computing. When a lot of users constantly change in access control, it will cause the authority to spend more computing resources. Therefore, how to avoid frequently change in the key update is an important issue. Security: the user's key will be a challenge because the key is generated by the authority. Because the key distribution is constructed in a secure channel, how to design a public channel scheme is an important issue.

References

[1] M. Armbrust, et al., "A view of cloud computing", *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] N. Attrapadung, H. Imai, "Conjunctive broadcast and attribute-based encryption", in *Proceedings of the 3rd International Conference on Pairing-Based Cryptography*, pp. 248–265, 2009.

[3] N. Attrapadung, H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes", in *Cryptography and Coding*, pp. 278–300, 2009.

[4] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP'07)*, pp. 321–334, California, USA, May 20-23, 2007.

[5] A. Boldyreva, V. Goyal, V. Kumar, "Identity-based encryption with efficient revocation", in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, pp. 417–426, 2008.

[6] V. Bovović, D. Socek, R. Steinwandt, V. I. Villányi, "Multi-authority attribute-based encryption with honest-but-curious central authority", *International Journal of Computer Mathematics*, vol. 89, no. 3, pp. 268–283, 2012.

[7] M. Chase, "Multi-authority attribute based encryption", in *Proceedings of the 4th Conference on Theory of Cryptography*, pp. 515–534, 2007.

[8] M. Chase, S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption.", in *Proceedings of the 16th ACM conference on Computer and Communications Security (CSS'09)*, pp. 121–130, 2009.

[9] V. Goyal, O. Pandey, A. Sahai, B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89–98, Alexandria, Virginia, USA, 2006.

[10] S. Guo, Y. Zeng, J. Wei, Q. Xu, "Attribute-based re-encryption scheme in the standard model", *Wuhan University Journal of Natural Sciences*, vol. 13, no. 5, pp. 621–625, 2008.

[11] J. Han, W. Susilo, Y. Mu, J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption", *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2150–2162, 2012.

[12] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, no. 6, pp. 98–115, 2015.

Table 7: Comparison of computation in the data re-encryption phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
CSS	$\wedge(T_{sym}+T_{Ge})$	$2T_{Ge}+T_h+T_B+\wedge(T_{Ge}+T_{GM})$	No	$2T_{Ge}+\wedge(2T_{Ge}+1T_h)$

Table 8: Comparison of computation in the data decryption phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
User	$\wedge(T_B+T_{Ge})$ $+T_B+T_{GM}$	$T_h+2T_B+T_{GM}(m+2)$ $+\wedge(T_{Ge}+T_B)+2T_{Ge}$	$\wedge(T_B+T_M)$ $+T_{GM}+T_{sym}$	$\wedge T_B+3T_{GM}+2T_B$ $+T_h+2T_{Ge}$

Table 9: Comparison of computation in the key update phase

	Hur and Noh [14]	Hur [13]	Yang et al. [40]	Zu et al. [45]
CSS	$5T_{Ge}+T_h+T_{GM}$ $+(\wedge-i)(2T_{Ge}+2T_{GM}$ $+T_h)+\wedge T_{sym}$	$6T_{Ge}+T_h+T_{GM}$ $+(\wedge-i)(2T_{Ge}+2T_{GM}$ $+T_h)+\wedge(T_{GM})$	$(\wedge-i)(T_{GM}+2T_{Ge})$	$2T_{Ge}+(\wedge-i)(6T_{Ge}$ $+T_{GM}+2T_h)$
Authority			$2T_{Div}+T_sub+2T_h$ $+4T_{Ge}+T_{Mul}+T_{GM}$	

[13] J. Hur, "Improving security and efficiency in attribute-based data sharing", *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2271–2282, 2013.

[14] J. Hur, D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[15] L. Ibraimi, M. Petkovic, S. Nikova, P. Hartel, W. Jonker, "Mediated ciphertext-policy attribute-based encryption and its application", in *Information Security Applications*, pp. 309–323, 2009.

[16] P. Junod, A. Karlov, "An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies", in *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, pp. 13–24, 2010.

[17] R. Kui, W. Cong, W. Qian, "Security challenges for the public cloud", *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[18] A. Lewko, B. Waters, "Decentralizing attribute-based encryption", in *Advances in Cryptology (EUROCRYPT'11)*, pp. 568–588, Springer Berlin Heidelberg, 2011.

[19] J. Li, X. Chen, C. Jia, W. Lou, "Identity-based encryption with outsourced revocation in cloud computing", *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 425–437, 2015.

[20] J. Li, Q. Huang, X. Chen, S. S. Chow, D. S. Wong, D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability", in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pp. 386–390, 2011.

[21] J. Li, K. Ren, K. Kim, "A2BE: Accountable attribute-based encryption for abuse free access control", *IACR Cryptology ePrint Archive*, vol. 20092009, pp. 118, 2009.

[22] J. Li, K. Ren, B. Zhu, Z. Wan, "Privacy-aware attribute-based encryption with user accountability", in *Information Security*, pp. 347–362, 2009.

[23] K. Liang, L. Fang, W. Susilo, D. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security", in *Proceedings of the 5th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS'13)*, pp. 552–559, 2013.

[24] X. Liang, Z. Cao, H. Lin, J. Shao, "Attribute based proxy re-encryption with delegating capabilities", in *Proceedings of the 4th ACM International Symposium on Information, Computer, and Communications Security (ASIACCS'09)*, pp. 276–286, 2009.

[25] X. Liang, R. Lu, X. Lin, X. S. Shen, *Ciphertext Policy Attribute Based Encryption with Efficient Revocation*, Technical Report, University of Waterloo, 2010.

[26] H. Lin, Z. Cao, X. Liang, J. Shao, "Secure threshold multi authority attribute based encryption without a central authority", *Information Science*, vol. 180, no. 13, pp. 2618–2632, 2010.

[27] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, T. H. Yuen, "Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles", in *Proceedings of the 16th European Con-*

ference on Research in Computer Security (ESORICS'11), pp. 278–297, Springer Berlin Heidelberg, 2011.

[28] S. Luo, J. Hu, Z. Chen, "Ciphertext policy attribute-based proxy re-encryption", in *Information and Communications Security*, pp. 401–415, 2010.

[29] D. Naor, M. Naor, J. Lotspiech, "Revocation and tracing schemes for stateless receivers", in *Advances in Cryptology (CRYPTO'01)*, pp. 41–62, Springer, 2001.

[30] R. OstrovOstrovsky, A. Sahai, B. Waters, "Attribute-based encryption with non-monotonic access structures", in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 195–203, 2007.

[31] M. Pirretti, P. Traynor, P. McDaniel, B. Waters, "Secure attribute-based systems", in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, pp. 99–112, 2006.

[32] S. Rafaeli, D. Hutchison, "A survey of key management for secure group communication", *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, 2003.

[33] A. Sahai, B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology (EUROCRYPT'05)*, LNCS 3494, pp. 457–473, Springer, 2005.

[34] H. J. Seo, H. Kim, "Attribute-based proxy reencryption with a constant number of pairing operations", *Journal of Information and Communication Convergence Engineering*, vol. 10, no. 1, pp. 53–60, 2012.

[35] J. Staddon, P. Golle, M. Gagne, P. Rasmussen, "A content-driven access control system", in *Proceedings of the 7th ACM Symposium on Identity and Trust on the Internet*, pp. 26–35, 2008.

[36] S. Subashini, V. Kavitha, "A survey on security issues in service delivery models of cloud computing", *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[37] Y. Wang, K. Chen, Y. Long, Z. Liu, "Accountable authority key policy attribute-based encryption", *Science China Information Sciences*, vol. 55, no. 7, pp. 1631–1638, 2012.

[38] X. Xie, H. Ma, J. Li, X. Chen, "New ciphertext-policy attribute-based access control with efficient revocation", in *Information and Communication Technology*, LNCS 7804, pp. 373–382, Springer, 2013.

[39] Z. Xu, R. Holloway, K. M. Martin, "Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage", in *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'12)*, pp. 844–849, 2012.

[40] K. Yang, X. Jia, *Security for Cloud Storage Systems*, Springer, New York, 2014.

[41] K. Yang, X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage", *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1735–1744, 2014.

[42] S. Yu, K. Ren, W. Lou, and J. Li, "Defending against key abuse attacks in KP-ABE enabled broadcast systems", in *5th International ICST Conference on Security and Privacy in Communication Networks (SecureComm'09)*, pp. 311–329, 2009.

[43] S. Yu, C. Wang, K. Ren, W. Lou, "Attribute based data sharing with attribute revocation", in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10)*, pp. 261–270, 2010.

[44] D. Zissis, D. Lekkas, "Addressing cloud computing security issues", *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[45] L. Zu, Z. Liu, J. Li, "New ciphertext-policy attribute-based encryption with efficient revocation", in *Proceedings of the 2014 IEEE International Conference on Computer and Information Technology (CIT'14)*, pp. 281–287, 2014.

Chih-Wei Liu received his M.S. in Soil And Water Conservation from National Chung Hsiung University, Taichung, Taiwan, ROC, in 2008. He is currently pursuing the Ph.D. degree from Computer Science and Information Engineering Department of Asia University, Taichung, Taiwan. His research interests include information security, cloud computing, and information law.

Wei-Fu Hsien received his B. S. in Department of Information Management from National Kaohsiung Marine University, Kaohsiung, Taiwan, ROC, in 2013. He is currently pursuing the M.S. degree with the Department of Management Information Systems from National Chung Hsing University. His research interests include security and privacy of cloud computing, and applied cryptography.

Chou-Chen Yang received his B.S. in Industrial Education from the National Kaohsiung Normal University, in 1980, and his M.S. in Electronic Technology from the Pittsburg State University, in 1986, and his Ph.D. in Computer Science from the University of North Texas, in 1994. From 1994 to 2004, Dr. Yang was an associate professor in the Department of Computer Science and Information Engineering, Chaoyang University of Tech- nology. Currently, he is a professor in the Department of Management Information Systems, National Chung Hshing University. His research interests include network security, mobile computing, and distributed system.

Min-Shiang Hwang received the B.S. in Electronic Engineering from National Taipei Institute of Technology, Taipei, Taiwan, Republic of China, in 1980; the M.S. in Industrial Engineering from National Tsing Hua University, Taiwan, in 1988; and the Ph.D. in Computer and Information Science from National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at National Cheng Kung University, Taiwan, from 1984-1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in

field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also a project leader for research in computer security at TL in July 1990. He obtained the 1997, 1998, and 1999 Distinguished Research Awards of the National Science Council of the Republic of China. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include database and data security, cryptography, image compression, and mobile communications.