Author: Jianping Yu, Weixin Xie, Jinyong Chen, Jianping Yu, Weixin Xie

Abstract—Ciphertext-policy attribute-based encryption (CP-ABE) is a very promising encryption technique for secure data sharing in the context of cloud computing. Data owner is allowed to fully control the access policy associated with his data which to be shared. However, CP-ABE is limited to a potential security risk that is known as key escrow problem whereby the secret keys of users have to be issued by a trusted key authority. Besides, most of the existing CP-ABE schemes cannot support attribute with arbitrary state. In this paper, we revisit attribute-based data sharing scheme in order to solve the key escrow issue but also improve the expressiveness of attribute, so that the resulting scheme is more friendly to cloud computing applications. We propose an improved two-party key issuing protocol that can guarantee that neither key authority nor cloud service provider can compromise the whole secret key of a user individually. Moreover, we introduce the concept of attribute with weight, being provided to enhance the expression of attribute, which can not only extend the expression from binary to arbitrary state, but also lighten the complexity of access policy. Therefore, both storage cost and encryption complexity for a ciphertext are relieved. The performance analysis and security proof show that the proposed scheme is able to achieve efficient and secure data sharing in cloud computing.

Index Terms—Secure data sharing, Attribute-based encryption, Removing escrow, Weighted attribute, Cloud computing.

I. INTRODUCTION

Cloud computing has become a research hot-spot due to its distinguished long-list advantages (e.g., convenience, high scalability). One of the most promising cloud computing applications is on-line data sharing, such as photo sharing in On-line Social Networks among more than one billion users [18], [22], [31] and on-line health record system [21], [26], [36]. A data owner (DO) is usually willing to store large amounts of data in cloud for saving the cost on local data management. Without any data protection mechanism, cloud service provider (CSP), however, can fully gain access to all data of the user. This brings a potential security risk to the user, since CSP may compromise the data for commercial benefits. Accordingly, how to securely and efficiently share user data is one of the toughest challenges in the scenario of cloud computing [1], [10], [17], [19], [20], [23], [25], [34], [38].

Ciphertext-policy attribute-based encryption (CP-ABE) [2], [4], [8], [12], [35], [39] has turned to be an important encryption technology to tackle the challenge of secure data sharing. In a CP-ABE, user’s secret key is described by an attribute set, and ciphertext is associated with an access structure. DO is allowed to define access structure over the universe of attributes. A user can decrypt a given ciphertext only if his/her attribute set matches the access structure over the ciphertext.

Employing a CP-ABE system directly into a cloud application that may yield some open problems. Firstly, all users’ secret keys need to be issued by a fully trusted key authority (KA). This brings a security risk that is known as key escrow problem. By knowing the secret key of a system user, the KA can decrypt all the user’s ciphertexts, which stands in total against to the will of the user. Secondly, the expressiveness of attribute set is another concern. As far as we know, most of the existing CP-ABE schemes [2], [4], [7], [8], [12], [15], [35], [37] can only describe binary state over attribute, for example, “1 - satisfying” and “0 - not-satisfying”, but not dealing with arbitrary-state attribute.

In this paper, the weighted attribute is introduced to not only extend attribute expression from binary to arbitrary state, but also to simplify access policy. Thus, the storage cost and encryption cost for a ciphertext can be relieved. We use the following example to further illustrate our approach.

Suppose there is a formal structure in university, in which teachers are classified into teaching assistant, lecturer, associate professor and full professor. We distribute the weight of the attribute for each type of the teachers as 1, 2, 3, and 4. Therefore, these attributes can be denoted as “Teacher: 1”, “Teacher: 2”, “Teacher: 3” and “Teacher: 4”, respectively. In this case, they can be denoted by one attribute which has just different weights. In particular, it can be arbitrary-state attributes, such as “Teacher: teaching assistant, lecturer, associate professor, full professor”. We here assume that an access policy is represented as $\mathcal{T} \{\langle \text{Lecturer} \rangle\text{ OR } \langle \text{Associate Professor} \rangle\text{ OR } \langle \text{Full Professor} \rangle\text{ AND Male}\}$, and the existing CP-ABE schemes are executed on the form of access policy $\mathcal{T}$. If our proposed scheme is deployed, the $\mathcal{T}$ can be simplified as $\mathcal{T} \{\langle \text{Teacher: 2} \rangle\text{ AND Male}\}$, since the attribute “Teacher: 2” denotes the minimum level in the access policy and includes (“Teacher: 2”, “Teacher: 3” “Teacher: 4”) by default. Therefore, the storage overhead of the corresponding ciphertext and the computational cost used in encryption can be reduced. These two structures are
shown in Fig. 1. In addition, our method can be used to express larger attribute space than ever under the same number of attributes. For example, if both the attribute space and weighted set include \( n \) elements, the proposed scheme can describe \( n^2 \) different possibilities. In contrast, the existing CP-ABE schemes only show \( 2n \) possibilities.

![Diagram](https://example.com/diagram1.png)

Fig. 1. Two equivalent access structures of a ciphertext. \( \mathcal{T} \) represents a general access policy in the existing CP-ABE schemes. \( \mathcal{T}' \) denotes an improved access policy in the proposed scheme.

A. Related Work

In 2005, Sahai and Waters [32] introduced fuzzy identity-based encryption (IBE), which is the seminal work of attribute-based encryption (ABE). After that, two variants of ABE were proposed: key-policy ABE (KP-ABE) [14] and CP-ABE [4, 8], depending on if a given policy is associated with either a ciphertext and a key. Later, many CP-ABE schemes with specific features have been presented in the literature. For example, [37] presented a novel access control scheme in cloud computing with efficient attribute and user revocation. The computational overhead is significantly eliminated from \( O(2N) \) to \( O(N) \) in user key generation by improving CP-ABE scheme, where \( N \) is the number of attributes. The size of ciphertext is approximately reduced to half of original size. However, the security proof of the scheme is not fully given.

Most of the existing CP-ABE schemes require a full trusted authority with its own master secret key as input to generate and issue the secret keys of users [4, 8, 13, 14, 27, 28, 32, 35, 37]. Thus, the key escrow issue is inherent, such that the authority has the “power” to decrypt all the ciphertexts of system users. Chase et al. [7] presented a distributed KP-ABE scheme to solve the key escrow problem in a multi-authority system. In this approach, all authorities, which are not colluded with each other, are participating in the key generation protocol in a distributed way, such that they cannot pool their data and link multiple attribute sets belonging to the same user. Because there is no centralized authority with master secret information, all attribute authorities should communicate with others in the system to create a user’s secret key. But, a major concern of this approach is the performance degradation [6, 30]. It results in \( O(N^2) \) communication overhead on both the system setup phase and any rekeying phase. It also requires each user to store \( O(N^2) \) additional auxiliary key components in addition to the attribute keys, where \( N \) is the number of authorities in the system. Chow [9] later proposed an anonymous private key generation protocol for IBE where a KA can issue private key to an authenticated user without knowing the list of the user’s identities. It seems that this approach can properly be used in the context of ABE if attributes are treated as identities. However, this scheme cannot be adopted for CP-ABE, since the identity of user is a set of attributes which is not publicly unknown.

In 2013, [15] provided an improved security data sharing scheme based on the classic CP-ABE [4]. The key escrow issue is addressed by using an escrow-free key issuing protocol where the key generation center and the data storage center work together to generate secret key for user. Therefore, the computational cost in generating user’s secret key increases because the protocol requires interactive computation between the both parties.

Besides, Liu et al. [27], [28] presented a fine-grained access control scheme with attribute hierarchy, where [27] and [28] are built on top of [8] and [35], respectively. In the schemes, the attributes are divided into multiple levels to achieve fine-grained access control for hierarchical attributes, but the attributes can only express binary state. Later, Fan et al. [13] proposed an arbitrary-state ABE to solve the issue of the dynamic membership management. In this paper, a traditional attribute is divided to two parts: attribute and its value. For example, the traditional attributes can be denoted as \{“Doctor”,“Professor”,“Engineer”\}. The improved attributes are denoted as: {Career: “Doctor”,“Professor”,“Engineer”}, where “Career” represents an attribute and “Doctor”,“Professor” and “Engineer” denote the values of the attribute “Career”. Accordingly, the computation cost for attributes is more expensive than that of the traditional schemes under the same number of attributes.

We note that there are some other research works on CP-ABE, such as [24], [25]. Nevertheless, they leverage different techniques to achieve data sharing. We will not compare them with our present system.

B. Our Contributions

Inspired by [37], we propose an attribute-based data sharing scheme for cloud computing applications, which is denoted as ciphertext-policy weighted ABE scheme with removing escrow (CP-WABE-RE). It successfully resolves two types of problems: key escrow and arbitrary-state attribute expression. The contributions of our work are as follows:

- We propose an improved key issuing protocol to resolve the key escrow problem of CP-ABE in cloud computing. The protocol can prevent KA and CSP from knowing each other’s master secret key so that none of them can create the whole secret keys of users individually. Thus, the fully trusted KA can be semi-trusted. Data confidentiality and privacy can be ensured.
- We present weighted attribute to improve the expression of attribute. The weighted attribute can not only express arbitrary-state attribute (instead of the traditional binary state), but also reduce the complexity of access policy. Thus the storage cost of ciphertext and computation complexity in encryption can be reduced. Besides, it can express larger attribute space than ever under the same condition. Note that the efficiency analysis will be presented in Section V.
we conduct and implement comprehensive experiment for the proposed scheme. The simulation shows that CP-WABE-RE scheme is efficient both in terms of computation complexity and storage cost. In addition, the security of CP-WABE-RE scheme is also proved under the generic group model.

II. PRELIMINARIES

A. Access Structure

Let \( \{P_1, \ldots, P_n\} \) be a set of parties. A collection \( \mathcal{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \) is monotone if for all \( B, C \); if \( B \in \mathcal{A} \) and \( B \subseteq C \) then \( C \in \mathcal{A} \). An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) \( \mathcal{A} \) of non-empty subsets of \( \{P_1, \ldots, P_n\} \), i.e., \( \mathcal{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \setminus \{\emptyset\} \). The sets in \( \mathcal{A} \) are called authorization sets. Otherwise, the sets are called unauthorization sets.

In our scheme, the role of the parties is taken by the attributes. Thus, \( \mathcal{A} \) is going to include the authorized sets of attributes. Generally, unless stated in another way, the scheme uses an access structure which is a monotone access structure.

B. Bilinear Mapping

Let \( G_0 \) and \( G_T \) be two multiplicative cyclic groups of prime order \( p \). The generator of \( G_0 \) is \( g \). A bilinear mapping \( \hat{\cdot} : G_0 \times G_0 \rightarrow G_T \) satisfies the following properties:

- Bilinearity: For any \( u, v \in G_0 \) and \( a, b \in \mathbb{Z}_p \), it has \( \hat{u}^a \cdot \hat{v}^b = \hat{(u \cdot v)}^{ab} \).
- Non-degeneracy: There exists \( u, v \in G_0 \) such that \( \hat{u}(v) \neq 1 \).
- Computability: For all \( u, v \in G_0 \), there is an efficient algorithm to compute \( \hat{u}(v) \).

C. Weighted Access Tree

Let \( T \) be a weighted access tree, where root node of the tree is \( R \). To facilitate description of the access tree, several functions and terms are defined as follows.

\[
\begin{align*}
\text{Index} & : k_x \\
\text{Attribute} & : \text{Teacher} \\
\text{Weight} & : \text{Seniority} \\
\text{Threshold} & : k_x \\
\end{align*}
\]

\[ \begin{array}{c}
\text{Manage} \\
\text{Upload} \\
\text{Access} \\
\text{Users}
\end{array} \]

**Fig. 2.** An example of weighted access tree structure.

- \( x \) denotes a node of tree \( T \). If \( x \) is a leaf node, it denotes an attribute with weight. If \( x \) is a non-leaf node, it denotes a threshold gate, such as “AND”, “OR” and “n-of-m \( n < m \)”. For example, the nodes \( C \) and \( A \) denote a threshold gate and an attribute respectively in Fig. 2.
- \( num_x \) denotes the number of \( x \)'s children in \( T \). For example, \( num_R = 2 \) in Fig. 2.
- \( k_x \) denotes threshold value of node \( x \), where \( 0 < k_x \leq num_x \). When \( k_x = 1 \) and \( x \) is a non-leaf node, it is an OR gate. When \( k_x = num_x \) and \( x \) is a non-leaf node, it is an AND gate. In particular, if \( x \) is a leaf node, \( k_x = 1 \). For example, \( k_R = 1 \) and \( k_C = 2 \) denote an OR gate and an AND gate respectively in Fig. 2.

**Fig. 3.** System model of CP-WABE-RE scheme in cloud computing.

**Key Authority (KA).** It is a semi-trusted entity in cloud system. Namely, KA is honest-but-curious, which can honestly perform the assigned tasks and return correct results. However, it will collect as many sensitive contents as possible. In cloud
system, the entity is responsible for the users’ enrollment. Meanwhile, it not only generates most part of system parameter, but also creates most part of secret key for each user.

Cloud Service Provider (CSP). It is the manager of cloud servers and also a semi-trusted entity which provides many services such as data storage, computation and transmission. To solve the key escrow problem, it generates both parts of system parameter and secret key for each user.

Data Owners (DO). They are owners of files to be stored in cloud system. They are in charge of defining access structure and executing data encryption operation. They also upload the generated ciphertext to CSP.

Users. They want to access ciphertext stored in cloud system. They download the ciphertext and execute the corresponding decryption operation.

\[ \text{KeyGen}(	ext{MSK}_1, S) \rightarrow (\text{SK}_1) \] KA inputs \text{MSK}_1 and a set of weighted attributes \( S \). It creates secret key \( \text{SK}_1 \) described by \( S \).

(2) In CSP, KeyGen, we propose an improved two-party key issuing protocol to remove escrow. KA and CSP perform the improved protocol with master secret keys of their own. Thus, none of them can create the whole set of secret keys for users individually. Meanwhile, we assume that KA does not collude with CSP since they are honest as in [16] (otherwise, they can obtain the secret keys of each user by sharing their master secret keys).

\[ \text{CSP.KeyGen}(	ext{MSK}_2) \rightarrow (\text{SK}_2) \] CSP inputs \text{MSK}_2 and the required information. It produces secret key \( \text{SK}_2 \) by executing the following key issuing protocol.

- **KeyCom**_{\text{KA} \rightarrow \text{CSP}}(\text{MSK}_1, ID_i, r, \text{MSK}_2) \rightarrow (\text{SK}_2).

It is an interactive algorithm between KA and CSP. KA inputs MSK_1, a user identity \( ID_i \) and a personalized secret \( r \). CSP inputs MSK_2 and \( ID_i \). At last, only CSP generates a personalized key component \( \text{SK}_2 \) for the corresponding user.

Then, the user constructs the whole secret key \( \text{SK} \) with the key components separately receiving from KA and CSP, i.e. \( \text{SK} = \{ \text{SK}_1, \text{SK}_2 \} \).

**Phase 4: Data Decryption.** This phase contains both algorithms: Users, Decrypt and Data, Decrypt. User first downloads file ciphertext \( E_{ck}(M) \) and content key ciphertext \( CT \) from CSP. If he satisfies conditions, he can get content key \( ck \) by calling Users, Decrypt algorithm. Then, he uses \( ck \) to further decrypt file \( M \) by using Data, Decrypt operation.

(1) **Users, Decrypt** (PP, SK, CT) \rightarrow (ck). User inputs PP, SK described by \( S \), and CT which includes access policy \( A \). Only when the weighted attribute set \( S \) matches the access policy \( A \), the content key \( ck \) is obtained.

(2) **Data, Decrypt** (\( E_{ck}(M), ck \)) \rightarrow (M). User inputs \( E_{ck}(M) \) and \( ck \). Based on symmetric decryption algorithm, it outputs file \( M \).

IV. THE PROPOSED CP-WABE-RE SCHEME

In this section, we present the construction of CP-WABE-RE system, including five procedures: system initialization, new file creation (data encryption), new user authorization (user key generation), data file access (data decryption), and data file deletion. In addition, the revocation scheme of [15] can be directly used in our proposed scheme. The reason is described as below. The revocation scheme is performed in the phase of data encryption. And the removing escrow is operated in the phase of user key generation. Therefore, in [15], the modification of removing escrow does not affect the use of revocation scheme since they are run in different phases.

A. System Initialization

Let \( G_p \) be a bilinear group of prime order \( p \), and \( g \) be a generator of \( G_p \). Let \( \hat{e} : G_p \times G_p \rightarrow G_T \) denote the bilinear map. Let \( H : \{0, 1\}^k \rightarrow G_p \) be a hash function. For any \( i \in Z_p \) and a set \( S = \{s_1, s_2, ..., s_m \in Z_p \} \), the Lagrange coefficient \( \Delta_{i,S} = \prod_{l \in S, l \neq i} (x - l)/(i - l) \). In addition, an universe of attribute set \( A = \{a_1, ..., a_n \} \) and a set of weights

![System framework of CP-WABE-RE scheme.](image-url)
\( W = \{\omega_1, \ldots, \omega_n\} (\omega_1 \leq \ldots \leq \omega_n) \) are defined. Thus the system contains \( n^2 \) weighted attributes which are \( \hat{A} = \{a_1 : \omega_1, \ldots, a_1 : \omega_n, \ldots, a_n : \omega_1, \ldots, a_n : \omega_n\} \), where the higher hierarchy of attributes is used, the bigger weighted value is distributed.

(1) **KA.Setup\( (1^k) \).** KA runs the algorithm which inputs a security parameter \( k \). Then, KA chooses random \( \alpha_1, \beta \in \mathbb{Z}_p \) and computes \( h = g^\beta \) and \( u_1 = \hat{e}(g, g)^{\alpha_1} \). Lastly, it obtains \( PP_1 \) and MSK1 as the formula (1):

\[
PP_1 = \{G_0, g, h, u_1\}, \quad MSK_1 = \{\alpha_1, \beta\}
\]

(2) **CSP.Setups\( (1^k) \).** CSP executes the operation which inputs a security parameter \( k \). Based on the \( k \), CSP chooses a random number \( \alpha_2 \in \mathbb{Z}_p \) and calculates \( u_2 = \hat{e}(g, g)^{\alpha_2} \). Then, it sets \( PP_2 \) and MSK2 as the formula (2):

\[
PP_2 = \{u_2\}, \quad MSK_2 = \{\alpha_2\}
\]

Finally, the public parameter and master secret key of system are denoted as \( PP = \{G_0, g, h, u = u_1 \cdot u_2 = \hat{e}(g, g)^{\alpha_2}\} \), where \( \alpha = \alpha_1 + \alpha_2 \), and MSK = \( \{\{\alpha_1, \beta\}, \{\alpha_2\}\} \).

**B. New File Creation (Data Encryption)**

Before file \( M \) is uploaded to CSP, DO processes the file with the following steps: (1) DO picks a unique ID for file \( M \). (2) It encrypts \( M \) with content key \( ck \) by using symmetric encryption method, where \( ck \) is chosen in a key space. The file ciphertext is denoted as \( E_{ck}(M) \), where \( E_{ck} \) denotes a symmetric encryption operation with the key \( ck \). (3) It defines an access structure \( T \) and encrypts the \( ck \) by running the improved encryption operation. Then, content key ciphertext CT is returned.

**DO.Encrypt(\( PP, ck, T \).** The improved algorithm is executed by DO which inputs \( PP, ck \) and \( T \). It outputs CT.

Firstly, a polynomial \( q_{c} \) is selected for each node \( x \) (including the leaf nodes) in \( T \). From the root node \( R \), the node’s information of \( q_{c} \) is randomly selected from top to bottom manner. For each node \( x \) in \( T \), degree of the polynomial \( d_x \) is set to \( k_x - 1 \), where \( k_x \) is the threshold value.

Then, beginning from the root node \( R \), DO sets \( q_{R}(0) = s(s \in \mathbb{Z}_p) \), where \( s \) is randomly selected. And DO randomly selects \( d_R \) other points of the polynomial \( q_R \) to define it completely. For each non-root node \( x \), it sets \( q_x(0) = q_{\text{parent}(x)}(\text{index}(x)) \) and randomly chooses \( d_x \) other points to completely define \( q_x \). Meanwhile, each leaf node denotes an attribute with weight.

In the access tree \( T \), let \( Y \) be the set of leaf nodes, and \( \omega_i \) be the minimum weight of each leaf node, which is set by DO. Then DO computes CT as the formula (3). Finally, DO sends the integrated ciphertext \( \{ID, CT, E_{ck}(M)\} \) to CSP.

\[
CT = \begin{cases}
T, \hat{C} = ck \cdot \hat{e}(g, g)^{\alpha_2}, \quad C = g^s \\
\forall y \in Y, i \in [1, n]: \\
C_y = h^{n_y(0)} \cdot H(\text{attr}(y))^{-\omega_y}, \\
\forall j \in (i, n], C_{y,j} = H(\text{attr}(y))^{-\omega_j}, \omega_j \in W)
\end{cases}
\]

To better understand, let us take the Fig. 2 for example here. DO sets the polynomial \( q_R(x) \) with degree \( 0 \) as \( q_R(x) = s \) for node \( R \). At the same time, the polynomial \( q_C(x) \) is set as \( q_C(x) = s + c_x \) with degree \( 1 \) where \( q_R(u) = q_C(0) = s \) and \( c_x \) is randomly chosen in \( \mathbb{Z}_p \) from node \( C \). In addition, for leaf nodes \( A, B \) and \( D \), they are respectively set as: \( q_A(0) = q_C(x) = s + c_x \), \( q_B(0) = q_C(y) = y \), and \( q_D(0) = q_R(z) = s \).

**C. New User Authorization (User Key Generation)**

When a user wants to join the cloud system, KA first accepts the user’s enrollment. If he is legal, KA authenticates and assigns a set of weighted attributes \( S \) to the user in accordance with his identity or role. Then, KA and CSP cooperate each other and generate secret key SK for the user. The phase consists of CSP.KeyGen and KA.KeyGen.

(1) **CSP.KeyGen.** We provide an improved key issuing protocol between KA and CSP to execute the work of CSP.

**KeyCon_{KA,CSP}(MSK_1, ID, r, MSK_2).** Assume that user \( t \) needs a secret key. Firstly, KA randomly chooses a number \( r \in \mathbb{Z}_p \) for the user. Then, KA and CSP perform a secure two-party computation (2PC) protocol, where KA inputs \( MSK_1 = \{\alpha_1, \beta\} \) and CSP inputs \( MSK_2 = \{\alpha_2\} \). After the execution of the 2PC protocol, CSP gets \( x = (\alpha_1 + \alpha_2)\beta \mod p \). This can be done via a general 2PC protocol for a simple arithmetic computation [7, 9, 15]. Alternatively, we can do this more efficiently by using the scheme in [3]. Note that during the 2PC protocol, KA knows nothing about \( \alpha_2 \) while CSP knows nothing about \( \{\alpha_1, \beta\} \). After the 2PC protocol, KA and CSP engage the following interactive protocol in order to generate the personalized key component for the user:

1. CSP selects a random number \( \rho_1 \in \mathbb{Z}_p \) and calculates \( X_1 = g^{\rho_1} = g^{(\alpha_1 + \alpha_2)\beta / \rho_1} \) and \( Y_1 = h^{\rho_1} \). It sends \( \{Y_1, Y_2, PoK(\theta, r)\} \) to KA.

2. KA chooses \( \theta \in \mathbb{Z}_p \) and computes \( X_2 = Y_1^{\rho_1} h^{\rho_2} = (g^{\alpha_1 + \alpha_2} h^{\beta / \rho_1})^{\rho_2} \). It transfers \( X_2, PoK(\rho_2) \) to KA.

3. CSP randomly selects \( \rho_2 \in \mathbb{Z}_p \) and computes \( X_2 = (Y_1^{\rho_1} Y_2)^{\rho_2} \). It transfers \( X_2, PoK(\rho_2) \) to KA.

4. KA calculates \( Y_3 = X_2^{1 / \rho_2} = g^{(\alpha_1 + \alpha_2)h^{\beta / \rho_1}} \). And sends \( \{Y_3, PoK(\theta)\} \) to CSP.

5. CSP calculates \( D = Y_3^{1 / \rho_2} = g^{(\alpha_1 + \alpha_2)h^{\beta / \rho_1}} \). And sends a personalized key component \( SK_2 = \{D = g^{\rho_1} h^{\rho_2}\} \) to the corresponding user \( t \).

Fig. 5 provides a direct description for the above protocol. Here the first step denotes a 2PC protocol which inputs \( \{\alpha_1, \beta\} \) from KA and \( \{\alpha_2\} \) from CSP, and returns \( x = (\alpha_1 + \alpha_2)\beta \mod p \) to CSP.

(2) **KA.KeyGen.** KA executes the algorithm which inputs MSK1, a number \( r \in \mathbb{Z}_p \) which has been randomly chosen in CSP.KeyGen, and a set of weighted attributes \( S \). Then, for each weighted attribute \( j \in S \), it possesses weighted value \( \omega_j \in W \). Finally, it computes SK1 described by S as the formula (4).
Fig. 5. The improved two-party key issuing protocol. “2PC” denotes a general two-party computation protocol. “PoK” denotes a proof of knowledge of the secret values used in the computation.

\[
\text{SK}_1 = \{L = g^r, \forall j \in S : D_j = H(j)^{\omega_j}\} \quad (4)
\]

So, user \( t \) can construct the whole secret key \( \text{SK} \) by using the key components separately receiving from the two entities. It is described as the formula (5).

\[
\text{SK} = \{D = g^r h^r, L = g^r, \forall j \in S : D_j = H(j)^{\omega_j}\} \quad (5)
\]

D. Data File Access (Data Decryption)

In cloud system, legal users can freely query the ciphertext. When a user requests CSP to access a ciphertext, it transmits the corresponding ciphertext \( \{ID, CT, E_{ck}(M)\} \) to the user. The user can obtain content key \( ck \) by calling the improved Users.Decrypt algorithm. Then, he uses \( ck \) to further decrypt the file \( M \) using Data.Decrypt operation.

1) Users.Decrypt(PP, CT, SK). User inputs PP, CT, and SK described by \( S \). If the weighted attributes \( S \) that the user possesses satisfy access policy \( T \), the user can obtain content key \( ck \). The operation is a recursive algorithm which is defined as below.

1) If \( x \) is a leaf node. Let \( k = \text{att}(x), \omega_k \) be the weighted value of the user’s node \( x \) and \( \omega_i \) be the weighted value of the access policy \( T \)’s node \( x \). If \( k \notin S \) or \( k \in S, \omega_i > \omega_k \), we note DecryptNode(CT, SK, x) \( = \perp \). If \( k \in S \) and \( \omega_i = \omega_k \), we compute DecryptNode(CT, SK, x) as the formula (6). If \( k \in S, \omega_i < \omega_k \) or \( \omega_k = \omega_i \), we compute DecryptNode(CT, SK, x) as the formula (7).

\[
\text{DecryptNode}(CT, SK, x)_1 = \hat{e}(C_x, L) \cdot \hat{e}(C, D_k)
\]

\[
= \hat{e}(h^{\alpha x(0)} \cdot H(\text{att}(x))^{-\omega_x}, g^r) \cdot \hat{e}(g^s, H(k)^{\omega_k})
\]

\[
= \hat{e}(g^{\beta x(0)} \cdot g^r) \cdot \hat{e}(H(k)^{-\omega_k}, g^r) \cdot \hat{e}(g^s, H(k)^{\omega_k})
\]

\[
= \hat{e}(g^s, H(k)^{\omega_k}) \quad (\text{if } \omega_k = \omega_i)
\]

2) If \( x \) is a non-leaf node, DecryptNode(CT, SK, x) is defined: for all nodes \( z \) that are children of \( x \), it runs DecryptNode(CT, SK, z) and stores the output as \( F_z \). Let \( S_z \) be an arbitrary \( k_z \)-sized set of child nodes \( z \) such that \( F_z \neq \perp \). If the nodes don’t exist, \( F_z = \perp \). If not, \( F_z \) is computed as the formula (8), where \( k = \text{index}(z) \), and \( S_z = \{\text{index}(z) : z \in S_z\} \).

\[
\begin{align*}
F_z = & \prod_{z \in S_z} F_{z, c}^{\Delta_k, s_z} \\
= & \prod_{z \in S_z} (\hat{e}(g, g)^{r \cdot \beta z(0)})^{\Delta_k, s_z} \\
= & \prod_{z \in S_z} (\hat{e}(g, g)^{r \cdot \beta z(0)})^{\Delta_k, s_z} \\
= & \hat{e}(g, g)^{r \cdot \beta z(0)}
\end{align*}
\]

Then, we define the decryption algorithm by calling DecryptNode(CT, SK, x) \( \perp \) or DecryptNode(CT, SK, x) \( \perp \) on the root node \( R \) of the access tree \( T \). If the \( T \) is satisfied by \( S \), we define \( A = \text{DecryptNode}_{\text{lookup}}(CT, SK, R) = \hat{e}(g, g)^{r \cdot \beta z(0)} = \hat{e}(g, g)^{\beta s} \). Thus, the user can gain \( ck \) with the formula (9).

\[
\hat{C}/\hat{e}(C, D)/A = \hat{C}/\hat{e}(g^s, g^\alpha \cdot h^r)/\hat{e}(g, g)^{r \cdot \beta s}
\]

\[
= ck \cdot \hat{e}(g, g)^{\alpha s}/\hat{e}(g, g)^{\alpha s}
\]

\[
= ck
\]

(2) Data.Decrypt(\( E_{ck}(M), ck \)). User inputs file ciphertext \( E_{ck}(M) \) and content key \( ck \). Based on symmetric decryption algorithm, i.e., DES or AES, the file \( M \) can be decrypted as the formula (10), where \( D_{ck} \) denotes a symmetric decryption operation with the key \( ck \).

\[
D_{ck}[E_{ck}(M)] = M
\]

E. Data File Deletion

Here, we show that data file deletion can perform both discretionary deletion and mandatory blocking.

Discretionary Deletion. All of the legal data owners can freely delete ciphertext in cloud system. Assume that a DO wants to delete an encrypted file, the procedures of algorithm between DO and CSP are described as below, where the algorithm can adopt any secure signature scheme such as BLS short signature scheme [5] as the underlying primitive to achieve.

1) DO sends a request to CSP, which includes file’s ID and its signature on the ID.

2) CSP verifies these request information. If validation, CSP deletes the corresponding ciphertext.

Mandatory Blocking. To provide legitimate aspect of file sharing, a new function, i.e., mandatory blocking, is added to the proposed system. The steps are described as below.
(1) When accessing a file, user needs to evaluate how well the file is accessed, such as shopping online and teaching online.

(2) CSP synthesizes these assessments for each file. If some files are not consistent with the evaluation standards, those files would be mandatory blocked by CSP. Meanwhile, DO will receive the private messages explaining the reason.

V. PERFORMANCE ANALYSIS

In this section, we analyze and compare the efficiency of the proposed scheme with the schemes [15], [37] and [13] in theoretical and experimental aspects.

A. Theoretical Analysis

1) Key Escrow and Weighted Attribute: Table I shows the problem of key escrow, feature of weighted attribute and application in cloud computing for each scheme. The key escrow in CP-WABE-RE scheme can be removed by using an improved key issuing protocol for cloud computing. [15] uses escrow-free key issuing protocol to solve the issue. On the contrary, both [37] and [13] don’t solve the problem of key escrow. In addition, the weighted attribute in CP-WABE-RE scheme can not only support arbitrary-state attribute instead of the traditional binary state, but also simplify access policy associated with a ciphertext as opposed to [15] [37].

Unfortunately, [13] can only express arbitrary-state attribute, and cannot simplify the access structure. In Table I, we can find that only CP-WABE-RE scheme can simultaneously support all the three functions. [15] solves the problem of key escrow so it can satisfy environment of cloud system as ours. However, both [37] and [13] cannot remove key escrow. Thus the both schemes cannot be directly applied in cloud computing.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Key Escrow</th>
<th>Weighted Attribute</th>
<th>Cloud System</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-WABE-RE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[15]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[37]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[13]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

2) Efficiency: In Table II and Table III, we compare efficiency of the above four schemes on storage overhead and computation cost in theory, where the used symbols are defined in Table IV.

In Table II, the schemes are compared in terms of CT size, SK size, PP size and MSK size. CT size represents the storage overhead in cloud computing and also implies the communication cost from DO to CSP, or from CSP to users. SK size denotes the required storage cost for each user. PP and MSK sizes represent the storage overhead of KA and CSP in terms of public parameter and master secret key.

As shown in Table II, when \( \omega_i = \omega_i^1 \), all attributes possess equal weights in CP-WABE-RE scheme. Thus our scheme is equivalent to [15] and [37]. Meanwhile, CT size is reduced as \((|A_C| + 1)L_{G_{0}} + L_{G_{T}}\) in CP-WABE-RE scheme, which is equal to [37]’s. Comparing with [15] and [13], the CT size in our proposed scheme and [37] is reduced by nearly half. When \( \omega_i \neq \omega_i^1 \), CP-WABE-RE scheme can use an attribute to express \( (\omega_i - \omega_i^1 + 1) \) attributes which have different weights. Therefore, it requires smaller storage cost in CT than the others. Moreover, we can find that the SK size in CP-WABE-RE scheme is equal to [37]’s, which is smaller than [15]’s and [13]’s. Furthermore, when \( |A_u| \rightarrow \infty \), the storage overhead in our scheme is reduced by nearly half comparing to [15]’s. And the storage cost in our scheme is decreased nearly by 66.67% comparing to [13]’s in theory. In addition, we can also observe that the PP size is equal among [15], [37] and CP-WABE-RE scheme. And the size of PP in [13] is the longest since it is related to the number of system attributes \( n \) and the number of system users \( k \). About the size of MSK, we can find that the parameter in CP-WABE-RE scheme doesn’t appear to be much different from the others.

In Table III, we evaluate the computation cost of encryption, decryption and user key generation. In the phase of new file creation (data encryption), the computation cost in CP-WABE-RE scheme can be reduced as \( 2|A_C| + 1)G_{0} + 2G_{T} \) when
\[ \omega_i = \omega_{i_1}, \] which is roughly equal to [15]'s, [37]'s and [13]'s. Similar to Table II, when \( \omega_i \neq \omega_{i_1} \), CP-WABE-RE scheme computes an attribute to represent multiple attributes which possess different weights. Meanwhile, it can simplify access structure associated with a ciphertext. However, the scheme [13] doesn’t possess the feature of our scheme, i.e., without expressing arbitrary-state attribute. So, when \( \omega_i \neq \omega_{i_1} \), the encryption cost in CP-WABE-RE scheme is saved. In the phase of file access (data decryption), the length of parameter is equal among [15], [37] and CP-WABE-RE scheme. And the computation cost on decryption in [13] is larger than the others. In addition, in the phase of new user authorization (user key generation), our proposed scheme only consumes additional \( 6G_0 \) of computation cost in solving key escrow issue, comparing with that in [37]. Meanwhile, the computation cost on key generation in CP-WABE-RE scheme is smaller than [15]'s and [13]'s. Furthermore, when \( |A_u| \rightarrow \infty \), the computation cost in ours is decreased nearly by 50% in theory than [15]'s, where the cost for transmission isn’t involved in both the two schemes. At the same time, the cost in ours is reduced by nearly 75% comparing to [13]'s in theory.

### B. Experimental Analysis

Now, to validate theoretical analysis proposed in previous subsection, we execute CP-WABE-RE scheme by using the cpabe toolkit and the Java Pairing-Based Cryptography library (JPBC) [11]. Meanwhile, we also simulate the schemes in [15], [37] and [13] at the same condition. The following experiments are conducted using Java on the system with Intel (R) Core(TM) i5-4590 CPU at 3.30 GHz and 8.00GB RAM running Windows 7. To achieve a 80-bit security level, the experiments use a 160-bit elliptic curve group based on the supersingular curve \( y^2 = x^3 + x \) over a 512-bit finite field. In addition, all the simulation results are the mean of 10 trials. The units of storage cost and time are Kilobyte (KB) and second (s).

#### 1) Simulation Analysis of Key Escrow

The storage overhead and computation cost of user secret key are compared as plotted in Fig. 6. The number of weighted attributes used in this simulation is \( N = \{10, 20, 30, 40, 50\} \).

Fig. 6(a) and Fig. 6(b) intuitively show the experimental results. We find that the storage overhead of user secret key in ours is the same as [37]'s, and it is smaller than [15]'s and [13]'s under the same number of attributes. About the computation cost of secret key, the value of our scheme is larger than [37]'s, where the difference is \( 6G_0 \) according to the Table III. At the same time, the parameter in CP-WABE-RE scheme is smaller than [15]'s and [13]'s at the same condition. We also observe that all experimental results are gradually increasing and approximately follow a linear relationship with the number of weighted attributes. Therefore, with a small error tolerance, we estimate their limit values, where the mathematical expressions are computed by using the mean algorithm. When \( N \rightarrow \infty \), the limit value of space saving in CP-WABE-RE scheme is approximately equal to 48.39% comparing to [15]'s. The cost is reduced by nearly half in theory which is consistent with the above efficiency analysis. Comparing with both our scheme and [13], the saved storage cost is approximately 64.47% which matches the corresponding limit value in theory. In addition, when \( N \rightarrow \infty \), comparing with CP-WABE-RE scheme and [15], the maximum of improved efficiency in computation cost approaches to 23.04%. Comparing with our scheme and [13], the reduced computation cost is approximate to 64.88% in Fig.

### TABLE II

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Size of CT</th>
<th>Size of SK</th>
<th>Size of PP</th>
<th>Size of MSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-WABE-RE</td>
<td>( \prod_{i=1}^{n} A_i \cdot (\omega_i - \omega_{i_1} + 1) + 1 ) ( L_{G_0} + L_{G_T} )</td>
<td>( (</td>
<td>A_u</td>
<td>+ 2)L_{G_0} )</td>
</tr>
<tr>
<td>[15]</td>
<td>( 2A_i + 1 ) ( L_{G_0} + L_{G_T} )</td>
<td>( 2A_i + 1 ) ( L_{G_0} + L_{G_T} )</td>
<td>( 3L_{G_0} + L_{G_T} )</td>
<td>( L_{G_0} + L_{G_T} )</td>
</tr>
<tr>
<td>[37]</td>
<td>( (</td>
<td>A_i</td>
<td>+ 1) ) ( L_{G_0} + L_{G_T} )</td>
<td>( (</td>
</tr>
<tr>
<td>[13]</td>
<td>( 2(</td>
<td>A_i</td>
<td>+ 1) ) ( L_{G_0} + 2L_{G_T} )</td>
<td>( 3</td>
</tr>
</tbody>
</table>

### TABLE III

<table>
<thead>
<tr>
<th>Scheme</th>
<th>New File Creation (Data Encryption)</th>
<th>Data File Access (Data Decryption)</th>
<th>New User Authorization (User Key Generation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP-WABE-RE</td>
<td>( \prod_{i=1}^{n} A_i \cdot (\omega_i - \omega_{i_1} + 2) ) ( + 1 ) ( G_0 + 2G_T )</td>
<td>( 2A_i + 1 ) ( C_e + (2</td>
<td>S</td>
</tr>
<tr>
<td>[15]</td>
<td>( 2A_i + 1 ) ( G_0 + 2G_T )</td>
<td>( 2A_i + 1 ) ( C_e + (2</td>
<td>S</td>
</tr>
<tr>
<td>[37]</td>
<td>( 2A_i + 1 ) ( G_0 + 2G_T )</td>
<td>( 2A_i + 1 ) ( C_e + (2</td>
<td>S</td>
</tr>
<tr>
<td>[13]</td>
<td>( 2A_i + 2 ) ( G_0 + 3G_T )</td>
<td>( 3A_i + 1 ) ( C_e + (2</td>
<td>S</td>
</tr>
</tbody>
</table>
equal weights. However, in Table III, if \(|A_n| \to \infty\), the corresponding computation costs can be reduced to 50% and 75% in theory, where the computation cost for transmission isn’t involved. Remarkably, the cost comes from the difference between theoretical value and experimental result.

2) Simulation Analysis of Weighted Attribute: Next, we measure and analyze the storage overhead and computation cost for encrypting (by a DO) data, where the number of attributes in access policy is \(N = \{10, 20, 30, 40, 50\}\). It should be noted that the CP-WABE-RE scheme is equivalent to the schemes in [15] and [37] when all attributes possess equal weights \((\omega_i = \omega_{i_1})\), where it has been analyzed in Table II and Table III. For simplicity, we have omitted the simulation when \(\omega_i = \omega_{i_1}\). To show the advantage of the weighted attribute here, in CP-WABE-RE scheme, the maximum value of each weighted attribute is set as 5, and the lowest value of each weighted attribute is chosen to be encrypted. We implement [15], [37], [13] and our proposed scheme under the equivalent access policy encrypted in ciphertext. Fig. 7 shows the simulation results.

Fig. 7(a) plots the relationship between the storage overhead of ciphertext and the number of weighted attributes in access policy. Fig. 7(b) shows encryption time of ciphertext versus the number of weighted attributes. When \(\omega_i \neq \omega_{i_1}\) (here we assume that an attribute can be represented 5 attributes which possess different weights), we find that CP-WABE-RE scheme requires less storage cost and encryption time than the others.
We also observe that all results approximately follow a linear relationship with the number of weighted attributes in access tree. Similar as the analysis of Fig. 6, we estimate the limit values with a small error tolerance, where the mathematical expressions are computed by using the mean algorithm. For example, in Fig. 7(a), when \( N \to \infty \), comparing with CP-WABE-RE scheme and [15], the limit value of space saving is approximately equal to 52.60\%. Similarly, the reduced storage cost in ciphertext is 11.77\% comparing our scheme to [37]. And comparing with [13], our proposed scheme can save storage cost approximate to 51.71\%. In addition, the reduced storage cost in [37] approaches to 46.28\% comparing to [15].

We first present the chosen plaintext attacks (CPA) security analysis of CP-WABE-RE scheme and shows public parameter PP to the adversary \( \mathcal{A} \).

1) **System Initi.** The challenger runs the operations of \texttt{KA.Setup} and \texttt{CSP.Setup} of CP-WABE-RE scheme and sends public parameter PP to the adversary \( \mathcal{A} \).

2) **Phase 1.** For the attribute sets \( S_1, \ldots, S_q (\forall i \in [1, \ldots, q]) \) chosen by \( \mathcal{A} \), he can repeatedly ask \( \mathcal{C} \) for the secret key \( SK \). Meanwhile, the challenger answers the secret key \( SK \) by running the algorithms of \texttt{CSP.KeyGen} and \texttt{KA.KeyGen}.

3) **Challenge.** \( \mathcal{A} \) submits two equal length messages \( M_0, M_1 \in \mathbb{G}_T \) and an access tree \( A \) to the challenger, where there should not be any secret key issued to \( \mathcal{A} \) such that the key satisfies \( A \). The challenger randomly picks a bit \( \mu \in \{0, 1\} \) and encrypts \( M_\mu \) with \( A \) by using the algorithm \texttt{DO.Encrypt}.

4) **Phase 2.** Same as the Phase 1 but with the restriction that the querying key cannot satisfy \( A \).

5) **Guess.** \( \mathcal{A} \) outputs a guess \( \hat{\mu} \) of \( \mu \). In this game, \( \mathcal{A} \) can win the game which is defined as \( \Pr[\hat{\mu} = \mu] - (1/2) \).

**Definition 2.** The proposed scheme is said to be secure against CPA if no probabilistic polynomial-time adversaries have non-negligible advantage in the above game.

We use the generic bilinear group model and the random oracle model to prove that no adversary can break the CPA security of our scheme with non-negligible probability. In other words, our security is reduced to mathematical properties of elliptic curve groups as well as security of target collision resistance hash function. We note that our security proof technique follows that of [4].

Consider two random encodings \( \xi_0, \xi_1 : \mathbb{F}_p \to \{0, 1\}^m \), where \( m > 3\log(p) \). We set \( \mathbb{G}_0 = \{\xi_0(x) : x \in \mathbb{F}_p\} \) and \( \mathbb{G}_T = \{\xi_1(x) : x \in \mathbb{F}_p\} \). In the security game, the simulator is given a random oracle for simulating hash function, and oracles in groups \( \mathbb{G}_0, \mathbb{G}_T \) and bilinear map \( e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T \) for computation queries. We are also given a random oracle to represent the hash function \( H \). And we refer to \( \mathbb{G}_0 \) as a generic bilinear group. Below, we give a lower bound on the advantage of a generic adversary in breaking the security of our scheme.

**Theorem 1.** For any adversary \( \mathcal{A} \), let \( q \) be a bound on the total number of group elements which \( \mathcal{A} \) receives from queries to the oracles for the hash function, groups \( \mathbb{G}_0, \mathbb{G}_T \) and bilinear map \( e \). In the security game, \( \mathcal{A} \) is required to tell if \( \tilde{C} \) is either \( \tilde{C} = \tilde{e}(g, g)^{\alpha} \) or \( \tilde{C} = \tilde{e}(g, g)^{\beta} \). It is not difficult to see that the modified game can be regarded as a hybrid argument in which the adversary is asked to tell \( \tilde{C} = \tilde{e}(g, g)^{\alpha} \) from \( \mathbb{G}_0 \) \( \tilde{C} = \tilde{e}(g, g)^{\beta} \) from \( \mathbb{G}_0 \). Accordingly, an adversary in the CP-ABE game with advantage \( \epsilon \) is transformed into an adversary \( \mathcal{A} \) in the modified game with advantage at least \( \frac{\epsilon}{2} \). Below we let \( g = \xi_0(1), g^x = \xi_0(x) \) and \( \tilde{e}(g, g)^{\mu} = \xi_1(x) \).

1) **System Initi.** The simulator chooses \( \alpha_1, \alpha_2, \beta \in \mathbb{F}_p \), and next sets \( h = g^\beta, u_1 = \tilde{e}(g, g)^{\alpha_1}, u_2 = \tilde{e}(g, g)^{\alpha_2} \), and \( \theta \in \mathbb{F}_p \). It further sends the PP \( \{g, h, u = u_1u_2\} \) to \( \mathcal{A} \).

2) **Hash Queries.** If \( \mathcal{A} \) issues a hash query on an attribute \( att(y) \), the simulator returns \( q^i \) and stores \( (t, att(y)) \) into \( \text{List}^H \), where \( t_i \in \mathbb{F}_p \).

3) **Key Queries.** Here we combine the simulations of the algorithms \texttt{CSP.KeyGen} and \texttt{KA.KeyGen} as one key query. We note that it will not bring additional advantage for \( \mathcal{A} \) in winning the game. When \( \mathcal{A} \) queries a user \( \hat{i} \)'s secret key for an attribute set \( S \), the simulator works as follows. It chooses \( r_i, w_j \in \mathbb{F}_p \), and computes \( D = g^{a\beta r_i}, L = g^\alpha, \forall j \in S : D_j = H(j)^{r_iw_j} \). The simulator finally sends the secret key to \( \mathcal{A} \) and stores \( (SK, t, S) \) into \( \text{List}^{SK} \).

4) **Challenge.** \( \mathcal{A} \) outputs two equal length messages \( M_0, M_1 \in \mathbb{G}_T \) and an access tree \( A \) to the simulator, where there should not be any secret key issued to \( \mathcal{A} \) such that the key satisfies \( A \). The simulator chooses a \( s \in \mathbb{F}_p \), and next uses linear secret sharing technique to construct shares \( \lambda_y \) of \( s \) for all attributes \( y \) in \( A \) as in the algorithm \texttt{DO.Encrypt}, where \( \lambda_y \) is uniformly and independently random in \( \mathbb{F}_p \), and \( \lambda_y \) can be seen as a linear combination of independent random variables (in \( \mathbb{F}_p \)) and \( s \). The simulator then chooses \( \theta \in \mathbb{F}_p \), and sets \( \tilde{C} = \tilde{e}(g, g)^{\alpha} \), \( C = g^\alpha, \forall y \in Y, i \in [1, n] \), \( C_y = g^{\lambda_y}H(att(y))^{-w_is} \), and \( \forall j \in (1, n) \), \( C_{y,j} = H(att(y))^{-w_is} \), where \( w_i, w_j \in \mathbb{F}_p \). The simulator sends the challenge ciphertext to \( \mathcal{A} \).
5) **Key Queries.** Same as the previous key queries phase but with the restriction that the querying key cannot satisfy \( A \).

6) **Guess.** \( A \) outputs a guess bit.

Below we consider unexpected collision. An oracle query can be seen as a rational function \( \theta = \eta/\psi \) in the variables \( \theta, \alpha, \beta, t, w, \lambda_1, s \). Suppose there are two distinct rational functions \( \theta = \eta/\psi \) and \( \theta' = \eta'/\psi' \). An unexpected collision event indicates that taking two different queries corresponding to the two functions, we have the same output due to random choice of variables. If the event happens, it means that \( \theta = \theta' \), and further \( \eta/\psi - \eta'/\psi' = 0 \). By the Schwartz-Zippel Lemma in [33], [40], the probability of the event is \( O(1/p) \). Therefore, the probability of a collision event is at most \( O(q^2/p) \). Accordingly, the unexpected collision will not occur in the simulations with probability \( 1 - O(q^2/p) \).

Remember that each group element is uniformly and independently chosen in the above simulations. \( A \) can tell the difference between \( \theta \) and \( as \) elements in \( G_T \) if there are two distinct queries \( \theta \) and \( \theta' \) leading to the same output. Assume \( \theta' = \gamma \theta \) and \( \theta = \gamma as \), we have \( \theta - \theta' = \gamma (as - \theta) \) such that \( \gamma \theta + \theta - \theta' = \gamma as \), where \( \gamma \) and \( \gamma' \) are non-zero constant. We will show that \( A \) cannot construct a query for \( \gamma as \) in \( G_T \).

We here observe all possible rational function queries in \( G_T \) by means of bilinear map and the group elements given to \( A \). It can be seen that \( A \) can obtain a transcript \( \{ g, g^3, g^5, g^{3\lambda_1}, g^{-t_1 w_1 s}, g^{-(w_j - w_i)} s t, g^a \beta s, g^r w t_i \} \) from one query. For another transcript, we set it as \( \{ g, g^3, g^5, g^{3\lambda_1}, g^{-t_1 w_1 s}, g^{-(w_j - w_i)} s t, g^a \beta s, g^r w t_i \} \). We first ignore \( g^3 \) since the elements with \( \alpha \) will be tagged with \( \beta \) which is irrelevant to \( as \). To output a factor \( as \), we should focus on elements with factors \( \alpha \) and \( s \). It is not difficult to see that there are three types of outputs with \( as \) in \( G_T \) from two transcripts. One is \( as + \beta rs \) (resp. \( as + \beta r's \)). To output \( as \), we need an element \( \beta rs \). However the element does not exist. The second format with \( as \) is \( -(w_j - w_i) t_i s t + (w_j - w_i) s t \beta r' \) (resp. \( -(w_j - w_i) t_i s t + (w_j - w_i) s t \beta r' \)). To eliminate the part right after \( + \), we need to concentrate on the elements with \( t_i \). The elements with \( \beta \lambda_i - t_i w_i s, \) and \( r w t_i \) fail to construct a cancel-out part as they are lack of a factor \( w_j \). For the element \( -(w_j - w_i) s t_i \), we need an element with \( \beta r' \) (resp. \( \beta r \)). But none of other elements satisfy our requirement. The last format with \( as \) is \( -(w_j - w_i) s t_i + \alpha \beta s \beta r' - t_i w_i s \beta r' + \alpha \beta \lambda_i \) (resp. \( -(w_j - w_i) s t_i + \alpha \beta s \beta r' - t_i w_i s \beta r' + \alpha \beta \lambda_i \)). If we cannot find elements to cancel out all of terms except for that of \( as \), it indicates \( A \) fails to construct \( \gamma as \). For simplicity, we only check with the last term \( \alpha \beta \lambda_i \). It can be seen that \( \alpha + \beta r \) is the only element with \( \alpha \). Thus, we need a \( \beta \lambda_i \) term. Nevertheless, the term does not exist.

From the above observation, we can therefore state that \( A \) fails to construct the query form \( \gamma as \).

In addition, an improved key issuing protocol is proposed to resolve the key escrow problem of CP-ABE in cloud computing. In this paper, we assume that they do not collude with each other to share their master secret keys. We say that our proposed key issuing protocol is secure when the following two aspects are satisfied. The first one is that the KA cannot derive the user secret key if the CSP is honest. The other is that the CSP cannot derive the user secret key while the KA is honest. Security analysis about the protocol is described as below.

**Theorem 2.** The proposed key issuing protocol in section IV-C is a secure protocol for computing \( g^a h^x \) with \( \beta = \alpha \) and \( r \) and \( \gamma \) are known to the trust parties. Assume that the underlying arithmetic 2PC and zero knowledge proofs are secure, and (for security against corrupt CSP) that DDH is hard.

**Proof.** First, to note that \( D = Y_1^{1/\rho} = X_1^{1/\rho} = (Y_1^{1/\rho} Y_2)^{1/\rho} = X_1^{1/\rho} h^x = g^{a + \alpha \beta r} = g^{a + \beta r}. \) To show the security we consider the cases of corrupting KA and corrupting CSP respectively.

1) For a corrupted KA, our simulator proceeds as follows:

Sim \_K: First, it will run the arithmetic 2PC simulator for computation of \( (\alpha_1 + \alpha_2) \beta \). In the process, it will extract \( \alpha_1 \). Next, the simulator will choose random values \( X_1 \in G_0 \), and send it to KA. It will receive \( Y_1 \) and \( Y_2 \) from the adversary KA, and two corresponding zero knowledge proofs. We will extract \( \beta \) and \( r \) from the corresponding proofs. Then, it will send \( \alpha_1, \beta \) and \( r \) to the trusted party, and receive \( g^{a + \alpha \beta r} \), which will be CSP's secret key output.

Consider a hybrid simulator \( Hyb \_K \) that takes as input of CSP's secret \( \alpha_2 \). It first runs the arithmetic 2PC simulator for the computation of \( x \) with the correct output value according to \( \alpha_2 \). Then the simulator completes the protocol as the honest CSP would do. This is clearly indistinguishable from the real CSP's protocol by the security of the arithmetic 2PC.

Now, assuming that the proof of knowledge scheme is secure, \( Hyb \_K \) should be indistinguishable from the above simulator \( Sim \_K \). This is because the value \( X_1 \) used by \( Sim \_K \) will be distributed identically to those in \( Hyb \_K \). (Since \( \rho_1 \) is chosen at random in the real protocol, \( X_1 \) will be distributed uniformly over \( G_0 \) in the real protocol as in the simulated protocol.) Thus, interaction with our simulation is indistinguishable from interaction with an honest CSP.

2) For a corrupted CSP, our simulator proceeds as follows:

Sim \_C: First, it will run the arithmetic 2PC simulator for computation of \( (\alpha_1 + \alpha_2) \beta \) using the arithmetic 2PC simulator. When the 2PC simulator provides \( \rho_2 \) and asks for output, it will correctly compute \((\alpha_2 + \alpha_1) \beta \). Then it will complete the execution as in the real protocol. This protocol is clearly indistinguishable from the real KA's protocol by security of the arithmetic 2PC.
In addition, we consider a second hybrid $Hyb'$ which is the same as the $Hyb_K$ to proceed the above protocol, but which uses the zero-knowledge simulator for all proofs of knowledge. This must be indistinguishable by the zero-knowledge property of the proof system. Here we only need show that the $Hyb_K$ is indistinguishable from the interaction with the above simulator.

Consider the reduction from DDH assumption: Given $g, A = g^a, B = g^b, C = g^c$, and we must decide whether $c = ab$ or $c \in \mathbb{Z}_p$. Here we define $X_1 = A = g^a$ and $h = g^a$ for $\sigma \in \mathbb{Z}_p$. As the $Sim_K$, we run the arithmetic 2PC simulator for computation of $(a + \sigma_2)\beta$ and extraction $\sigma_2$. Next we receive $X_1 = A$ and extract $\rho_1$ from the corresponding proof. Meanwhile, we compute $Y_1 = X_1^{\beta} = g^{ab} = C^\beta$, $Y_2 = h^{\sigma_1} = g^{\sigma_1\sigma} = B^{\sigma_1}$, and send them to the adversary CSP, along with a simulated proof of knowledge. Then we receive $X_2$ and extract $\rho_2$ from the proof. At last, we compute $Y_3 = X_2^{1/\beta} = (A^{\rho_1/\beta} g^{\sigma_1})^{\rho_2}$ and send it to CSP.

Here we assume that the proofs of knowledge are secure. If $c = ab$, $Y_1$, $Y_2$, $Y_3$ will be distinguished correctly, and this will be distinguishable from $Hyb_K$. If $c$ is a random number (that is, $c \in \mathbb{Z}_p$), then $Y_1$, $Y_2$ are randomly selected from $\mathbb{Z}_p$, as in $Sim_K$. Thus, any adversary that can distinguish $Hyb_K$ from $Sim_K$ will allow us to resolve DDH problem. Under the DDH assumption, interaction with $Sim_K$ is indistinguishable from interaction with a real KA.

VII. CONCLUSION

In this paper, we redesigned an attribute-based data sharing scheme in cloud computing. The improved key issuing protocol was presented to resolve the key escrow problem. It enhances data confidentiality and privacy in cloud system against the managers of KA and CSP as well as malicious system outsiders, where KA and CSP are semi-trusted. In addition, the weighted attribute was proposed to improve the expression of attribute, which can not only describe arbitrary-state attributes, but also reduce the complexity of access policy, so that the storage cost of ciphertext and time cost in encryption can be saved. Finally, we presented the performance and security analyses for the proposed scheme, in which the results demonstrate high efficiency and security of our scheme.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (61171072, 61472083 and 61170283), in part by the Science and Technology Projects of Shenzhen (ZDSYS20140430164957660 and JCYJ20140418095735608), in part by the National High-Technology Research and Development Program (“863” Program) of China under Grant 2013AA01A212, and in part by the privacy-aware retrieval and modelling of genomic data (PRIGENDA, No. 13283250), the Academy of Finland.

REFERENCES


**Shulan Wang** is an MD-PhD student in the College of Information Engineering, Shenzhen University, China. She received the B.S. degree in electronic information engineering from Henan Polytechnic University, China, in 2010. Her research interests include cryptography, cloud computing, and information security.

**Kaitai Liang** is currently a Post Doctoral Researcher with the Department of Computer Science, Aalto University, Finland. He received the Ph.D. degree from the Department of Computer Science, the City University of Hong Kong, China, in 2014. His research interests are cyber-security, privacy and security in information technology; in particular, big data security, privacy enhancing technology, economic privacy, cloud security, privacy in Internet of Things and lightweight secure systems.

**Joseph K. Liu** is a senior lecturer at Faculty of Information Technology, Monash University, Australia. He got the Ph.D. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, China, in 2004. His current technical focus is particularly cyber security in the cloud computing paradigm, smart city, lightweight security, and privacy enhanced technology. He has published more than 90 referred journal and conference papers and received the Best Paper Award from ESORICS 2014 and ESORICS 2015. He has served as the program chair of ProvSec 2007, 2014, ACISP 2016 and as the program committee of more than 35 international conferences.

**Jianying Chen** is a professor in College of Computer Science and Software Engineering, Shenzhen University. He got the Ph.D. degree from City University of Hong Kong, Hong Kong, China, in 2003. He is interested in Artificial Intelligence and Information Security. He worked for ZTE Corporation as senior engineer of network technology from 2003 to 2006. After that, he joined the Shenzhen University. He was vice-chairman of International Telecommunication Union-Telecommunication (ITU-T) SG17 from 2004 to 2012, and editor of three recommendations developed in ITU-T SG17. He has published more than 30 papers and got more than 30 patents in the field of Artificial Intelligence and Information Security.

**Jiaping Yu** is a professor of College of Information Engineering, Shenzhen University, China. He got the M.S. degree in communication and electronic system from Northwestern Polytechnical University, China, in 1992, and the Ph.D. degree in communication and electronics at Xidian University, China, in 1995. His main research interests include cryptography, information and communication security, network security, internet of things, and cloud computing. He has published over 100 academic research papers.

**Weixin Xie** is professor and doctoral tutor in College of Information Engineering, Shenzhen University, China. He is currently the commissioner of the Chinese Institute of Electronics Signal Processing Branch. His main research interests include signal processing, intelligent information processing, radar target recognition, THz-TDS signal and image analysis. He has published over 200 academic research papers.