The Effect of Isomorphic Pitch Layouts on the Transfer of Musical Learning
Unified Ciphertext-Policy Weighted Attribute-Based Encryption for Sharing Data in Cloud Computing

Wei Li 1,*, Wei Ni 2, Dongxi Liu 2, Ren Ping Liu 3 and Shoushan Luo 1

1 National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; buptlou@263.net
2 Data61, CSIRO, Sydney, NSW 2122, Australia; wei.ni@data61.csiro.au (W.N.);
dongxi.liu@data61.csiro.au (D.L.)
3 Global Big Data Technologies Centre, University of Technology Sydney, Sydney, NSW 2007, Australia;
renping.liu@uts.edu.au
* Correspondence: weilihero@bupt.edu.cn; Tel.: +86-152-1019-4337

Received: 6 November 2018; Accepted: 4 December 2018; Published: 6 December 2018

Abstract: With the rapid development of cloud computing, it is playing an increasingly important role in data sharing. Meanwhile, attribute-based encryption (ABE) has been an effective way to share data securely in cloud computing. In real circumstances, there is often a mutual access sub-policy in different providers’ access policies, and the significance of each attribute is usual diverse. In this paper, a secure and efficient data-sharing scheme in cloud computing, which is called unified ciphertext-policy weighted attribute-based encryption (UCP-WABE), is proposed. The weighted attribute authority assigns weights to attributes depending on their importance. The mutual information extractor extracts the mutual access sub-policy and generates the mutual information. Thus, UCP-WABE lowers the total encryption time cost of multiple providers. We prove that UCP-WABE is selectively secure on the basis of the security of ciphertext-policy weighted attribute-based encryption (CP-WABE). Additionally, the results of the implementation shows that UCP-WABE is efficient in terms of time.

Keywords: attribute-based encryption; cloud computing; data sharing

1. Introduction

As one of the most promising applications, cloud computing [1–3] provides a more efficient way for data sharing. It enables data providers to store their data remotely in a cloud, and once data consumers can access the cloud, they can access the data any time and anywhere. Despite cloud computing supplying great convenience for data sharing, it also brings the serious challenge of information security [4]. Massive data are stored in the cloud storage platforms, and the data often contain sensitive information, such as personal health records in the medical cloud [5–7] and banking transactions in the financial cloud [8–10]. An untrustworthy entity named the cloud service provider (CSP) runs the cloud storage platforms, and it may steal the sensitive information to make a profit. Therefore, how to enforce a secure and efficient data sharing in the cloud has attracted many scholars’ attention [11–14].

Attribute-based encryption (ABE) [15] is a widespread cryptographic technology to protect the security of data in cloud computing. Different from traditional public key encryption, ABE does not encrypt plaintexts for an explicit consumer. Consumers’ secret keys and ciphertexts are associated with a set of attributes or an access policy, respectively. A consumer can decrypt a ciphertext if and only if his/her secret key has a match with the ciphertext. Weighted attribute-based
encryption [16–21] is a variant of ABE. It assigns different weights to attributes according to their importance. For example, we can give full professor and distinguished professor with weights one and two, denoted as “professor(1)” and “professor(2)”, respectively. This can avoid a very complicated access structure and improve the efficiency of the encryption in certain scenarios.

1.1. Problem Statement

In practical applications [7,22], different data providers share their own data with weighted access policies, and these policies may have a mutual sub-policy. A simple instance is given to elucidate this issue.

As illustrated in Figure 1, the data providers have data $M_1$ and $M_2$ to share separately. The weighted access policy of $M_1$ is $\Gamma_1\{\text{Attr3}(3) \text{ AND } (\text{Attr1}(2) \text{ AND } \text{Attr2}(3))\}$. The weighted access policy of $M_2$ is $\Gamma_2\{\text{Attr4}(2) \text{ AND } (\text{Attr1}(2) \text{ AND } \text{Attr2}(3))\}$. We notice that there is a mutual sub-policy $\{\text{Attr1}(2) \text{ AND } \text{Attr2}(3)\}$ between $\Gamma_1$ and $\Gamma_2$, and this means that the encryptions of $M_1$ and $M_2$ have some processes of repetition. This provides the possibility of improving the efficiency of the data sharing.

![Figure 1](image)

Figure 1. Two data providers have data $M_1$ and $M_2$ to share separately. $\Gamma_1$ and $\Gamma_2$ are the weighted access policies of $M_1$ and $M_2$.

1.2. Our Contributions

In this paper, we propose a new data-sharing scheme, which is called unified ciphertext-policy weighted attribute-based encryption (UCP-WABE), in cloud computing. UCP-WABE achieves both security and high performance. In this scheme, every attribute has its own weight depending on its importance defined in the system. The data providers share their data under their weighted access policies. When there is a mutual sub-policy among the weighted access policies, UCP-WABE optimizes the encryption of the data. UCP-WABE is proven to be selectively secure [23] on the basis of CP-WABE [16]. We carry out experiments for UCP-WABE, and the implementation results exhibit that UCP-WABE has better efficiency.

1.3. Organization

The rest of this paper is arranged as follows. Section 2 describes related work. Section 3 introduces preliminaries. In Section 4, we present the system model and security model of UCP-WABE. Section 5 proposes our data-sharing scheme. Section 6 represents the security analysis of UCP-WABE. The implementation results are presented in Section 7. Finally, the conclusions are stated in Section 8.

2. Related Work

As a popular cryptographic primitive used in cloud computing, ABE has two categories. One is ciphertext-policy ABE (CP-ABE) [24], and the other is key-policy ABE (KP-ABE) [25]. The major difference between CP-ABE and KP-ABE is the relationship of the ciphertext and secret key with the access policy and attributes. Specifically, for the former, a ciphertext is associated with an access policy that is expressed by threshold gates and attributes. A consumer’s secret key has attributes
embedded. For the latter, a ciphertext is associated with attributes, and an access policy is embedded into a consumer’s secret key. CP-ABE enables data providers to construct its access policy freely; in other words, the providers describe who can get the shared data flexibly. Therefore, CP-ABE is more suitable for data sharing in the cloud.

In order to increase the efficiency and enlarge the use scope of ABE for data sharing, many researchers have made great efforts. Liu et al. [17] introduced the concept of weight into CP-ABE and proposed a ciphertext-policy weighted attribute-based encryption scheme. In this scheme, the authority transforms the attribute set into the weight attribute separation set, then the data provider encrypts the data with linear secret sharing methods [26]. Although the size of the ciphertext and private key increases, the scheme achieves fine-grained access control and reflects the significance of attributes. Therefore, the scheme is more suitable for the practical applications. However, the scheme only supports the linear secret sharing scheme (LSSS) [27] access structure. On the basis of [17], Liu et al. [18] proposed another ciphertext-policy weighted attribute-based encryption scheme that supports threshold access structure [28]. This structure can consist of a threshold and many attributes. Therefore, this scheme is more expressive and is more appropriate for the cloud computing environment. Meanwhile, the scheme is proven secure under the selective-weighted attribute model. Nevertheless, the scheme only supports the threshold access structure, which only contains one threshold gate in one structure, and this still limits the expression. Wang et al. [19] proposed a multi-authority-based weighted attribute encryption scheme, which introduced the concept of weight into multi-authority-based attribute encryption [29]. In this scheme, a central authority assigns a unique user identifier (UID) to each consumer. The UID is the global identity of a consumer in the system, and it is used to generate the attribute secret keys issued by multiple authorities. The attribute authorities assign different weights to attributes depending on their importance; whereas, the scheme only supports the threshold access structure. Liu et al. [16] proposed an improved ciphertext-policy weighted attribute-based encryption (CP-WABE) to improve the efficiency over the traditional CP-ABE scheme. In this scheme, every attribute has its own weight according to its importance. The data provider encrypts its data under a weighted access policy. The data consumer can decrypt the ciphertext only if his/her attributes satisfy the weighted access policy. The scheme supports the tree access structure, so it can encrypt data under a more complex access policy. The scheme is proven to be secure under the decision $^\ell$-Expanded bilinear Diffie–Hellmann exponent ($^\ell$-Expanded BDHE) assumption [30]. Ghosh et al. [21] proposed a secure and efficient data collaboration scheme, which is called blowfish hybridized weighted attribute-based encryption. In this scheme, The weight is assigned to each attribute based on its importance, and data are encrypted using access control policies. The consumers can access the data corresponding to their weight in order to reduce the computational overload. Nevertheless, when there is a mutual sub-policy among the access policies of providers, all above-mentioned schemes do not consider further optimization.

3. Preliminaries

In this section, we introduce the basic concepts of bilinear mapping [31] and the weighted access tree [16].

3.1. Bilinear Mapping

Let $G_1$ and $G_2$ be two cyclic groups of prime order $p$ with the multiplication. Let $g$ be a generator of $G_1$ and $\hat{e}$ be a bilinear mapping. Let $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a bilinear mapping having the following properties:

- Bilinearity: For any $u, v \in G_1$ and $a, b \in \mathbb{Z}_p$, $\mathbb{Z}_p$ is the set of integers $[0, p-1]$. It has $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Non-degeneracy: $\hat{e}(g, g) \neq 1$.
- Computability: For all $u, v \in G_1$, there is an efficient computation $\hat{e}(u, v)$.

Note that $\hat{e}$ is symmetric since $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab} = \hat{e}(u^b, v^a)$. 

3.2. Weighted Access Tree

A weighted access tree is an image representation of a weighted access policy. Let \( \Gamma \) be a weighted access tree. Let \( N_j \) be the nodes of \( \Gamma \) and \( N_1 \) be the root of \( \Gamma \). There are two kinds of nodes in \( \Gamma \), non-leaf node and leaf node. The former represents a threshold gate, for example “AND”, “OR”, “\( a \leq b \) (\( a < b \))”. \( num_j \) and \( V_j \) describe the threshold gate, where \( num_j \) denotes the number of children of \( N_j \) and \( V_j \) denotes the threshold value, \( 0 < V_j \leq num_j \). When \( V_j = num_j \), the threshold gate is an AND gate, and when \( V_j = 1 \), it is an OR gate. A leaf node is described by a data consumer’s attribute with weight, and \( V_j = 1 \). We denote the parent of \( N_i \) by \( \text{parent}(N_i) \) and denote the index for \( N_i \) in its parent’s children by \( \text{index}(N_i) \), where \( 1 \leq \text{index}(N_i) \leq num_i \).

In \( \Gamma \), every non-leaf node \( N_j \) corresponds to a function \( V_j/num_j \). \( V_j/num_j \) is a Boolean function, and it is TRUE if \( N_j \) has at least \( V_j \) child nodes whose Boolean functions are evaluated to be TRUE. We say that a data consumer’s attribute set \( S \) satisfies \( \Gamma \) if \( \Gamma(S) = \text{TRUE} \), which is defined as described below.

For any leaf node \( N_j \) that is associated with an attribute \( a_i \in S \), if its Boolean value is TRUE, the weight of \( a_i \) from \( S \) must be greater than or equal to the weight of \( N_j \). For any non-leaf node, its Boolean value is the value of its Boolean function. If and only if the tree’s root node’s Boolean value is TRUE, then \( \Gamma(S) = \text{TRUE} \).

For instance, as shown in Figure 1, two weighted access trees correspond to two weighted access policies. \( \Gamma_1 : \{\text{Attr3}(3) \text{ AND } (\text{Attr1}(2) \text{ AND Attr2}(3))\} \), \( \Gamma_2 : \{\text{Attr4}(2) \text{ AND } (\text{Attr1}(2) \text{ AND Attr2}(3))\} \). Considering a data consumer whose attribute set \( S \) is \{\text{Attr1}(3), \text{Attr2}(3), \text{Attr3}(4), \text{Attr4}(1)\}. We could calculate that \( \Gamma_1(S) = \text{TRUE} \), \( \Gamma_2(S) = \text{FALSE} \). Table 1 gives several outcomes of \( \Gamma(S) \) with different attribute sets \( S \).

<table>
<thead>
<tr>
<th>( S )</th>
<th>( \Gamma_1(S) )</th>
<th>( \Gamma_2(S) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr1(3), Attr2(2), Attr3(4), Attr4(2)</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Attr1(3), Attr2(3), Attr3(4), Attr4(2)</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>Attr1(3), Attr2(3), Attr3(4), Attr4(1)</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>Attr1(3), Attr2(3), Attr3(4), Attr4(2)</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

4. System Model and Security Model of UCP-WABE

In this section, we describe the system model and security model of UCP-WABE. The system model shows how UCP-WABE enforces data sharing in cloud computing. The security model makes a foundation for the security analysis in Section 6.

4.1. System Model of UCP-WABE

Figure 2 displays the data-sharing system using UCP-WABE in the cloud environment. This system model is improved from the system model of CP-WABE [16]. The system model consists of five types of entities: a weighted attribute authority (WAA), a mutual information extractor (MIE), a cloud storage platform, numerous data providers, and data consumers. The cloud storage platform is managed by CSP and offers data storage service due to its massive storage ability. The provider’s data are encrypted with its weighted access policy and uploaded to the cloud. The data consumer downloads the ciphertexts and decrypts them with his/her secret key to recover the data. All providers and consumers are supervised by WAA. WAA is responsible for authenticating the attributes of every data consumer. WAA also assigns different weights to attributes according to their importance. The MIE assists the providers in encrypting their data. For illustration, we consider two providers in the system.
In Figure 2, WAA is a trustworthy entity that is in charge of producing the data consumers’ secret keys. CSP is honest but curious. This means that the cloud implements the manipulations dutifully and gives true outcomes; whereas, it also tries its best to get sensitive information. A data consumer could be compromised by an adversary; thus, this consumer could make attempts at obtaining data beyond his/her access range. The communication channels among the providers, the consumers, as well as the cloud are unsafe. In other words, the data in these channels could be eavesdropped by the adversary.

4.2. Security Model of UCP-WABE

The security model of UCP-WABE is described by a selective security game between a challenger and an adversary. This game is borrowed from CP-WABE [16]. The phases of the game are the following:

Init: The adversary declares the challenge weighted access policy \( \Gamma^* \) that he/she will try to attack and sends \( \Gamma^* \) to the challenger.

Setup: Here, the challenger calls the \texttt{Setup} algorithm to generate and send the public parameters to the adversary.

Phase 1: In this phase, the adversary can adaptively query for secret keys for the weighted attribute sets \( S_{q_1}, S_{q_2}, \ldots \) to the challenger. The challenge weighted access policy \( \Gamma^* \) must not be satisfied by any one of the queried attribute sets. The challenger responds by running the \texttt{KeyGen} algorithm to generate the corresponding secret keys \( SK_{q_1}, SK_{q_2}, \ldots \).

Challenge: The adversary submits two messages of the same length \( m_0, m_1 \). Then, the adversary makes a weighted access policy \( \Gamma' \) that there is a mutual access sub-policy between \( \Gamma^* \) and \( \Gamma' \). \( \Gamma' \) is sent to the challenger, as well. The challenger flips a random coin \( b \leftarrow \{0, 1\} \) and encrypts \( m_b \) under \( \Gamma^* \). The ciphertext \( CT^* \) is given to the adversary.

Phase 2: This phase is the same as Phase 1. The adversary asks for more secret keys, and the same restriction is that \( \Gamma^* \) must not be satisfied by any one of the queried attribute sets.

Guess: The adversary outputs his/her guess \( b' \) on \( b \).

Definition 1. A UCP-WABE scheme is selectively secure if all polynomial time adversaries have at most a negligible advantage in the above security game, where the advantage of an adversary is defined as
\[
\text{Adv} = \Pr[b' = b] - (1/2) [32].
\]

According to this definition, if we prove that there is no such polynomial time adversary who has a non-negligible advantage in the above security game, we can prove that UCP-WABE is selectively secure. The details of the proof will be described in Section 6.1.

5. Proposed Unified Ciphertext-Policy Weighted Attribute-Based Encryption

This section first gives an overview of our scheme. Then, we articulate the proposed UCP-WABE data-sharing scheme, which consists of five algorithms.
5.1. Overview of UCP-WABE

In UCP-WABE, different providers’ weighted access trees can be merged into one if and only if the trees meet two conditions. One is that there is a mutual sub-policy between the weighted access trees. The other is that the sub-trees’ roots of every weighted access tree locate in a trunk. A merging process is displayed in Figure 3. The two weighted access policies belong to different providers and have a mutual sub-policy: \{Attr3(3) AND (Attr1(2) OR Attr2(3))\}. All roots of the left weighted access tree are in the trunk: “AND”–“AND”–“OR”. All roots of the right weighted access tree are in the trunk: “OR”–“AND”–“OR”. Therefore, the two conditions are met, and the weighted access trees can be merged. The structure of the new weighted access tree is a multi-root tree.

5.2. The Design of UCP-WABE

UCP-WABE can be described as a collection of the following five algorithms:

**Setup**($\lambda, U$). This algorithm generates the public parameters ($PP$) and the master secret key ($MSK$). WAA runs **Setup**, which takes a security parameter $\lambda$ and the set of all attributes $U$ in the system as input. Each attribute has a weight depending on its importance in the system, and let $l_i$ be the maximum weight of attribute $u_i$, where $u_i \in U$.

WAA chooses a group $G_1$ of prime order $p$ with generator $g$ and random group elements \{${h_i, h_{i,j}}$\} $\forall u_i \in U, j = 0, \ldots, l_i$. WAA also chooses a random exponent $\alpha \in \mathbb{Z}_p$ and computes $PP$ as:

$$PP = \{G_1, g, \hat{e}(g, g)^{\alpha}, \{h_i, h_{i,j}\} \forall u_i \in U, j = 0, \ldots, l_i\}.$$ (1)

The master key of the system $MSK$ can be calculated as:

$$MSK = g^{-\alpha}.$$ (2)

$PP$ is released to all the data providers and data consumers.

**MutualInfoGen**($PP, \Gamma_1, \Gamma_2$). In [24], the cpabe toolkit [33] can transform an access policy into a stylized expression. Therefore, MIE is able to extract the mutual sub-policy when providers’ access policies meet the two conditions. Then, MIE creates an access tree $\Gamma_e$ according to the mutual sub-policy. Let $Y_e$ denote the set of the leaf nodes in $\Gamma_e$. Each element of $Y_e$ represents an attribute (denoted by $a_i$) with a weight (denoted by $\omega_i$). The number of leaf nodes is denoted by $k_e$. $\forall a_i \in Y_e$. Let $(i, j)$ denote the $j$th part of $a_i$, where $j = 1, \ldots, \omega_i$. MIE chooses a polynomial $f_e(x)$ for every node of $\Gamma_e$. Let the polynomial degree $d_e = V_e - 1$, as follows:

$$f_e(x) = \sum_{i=0}^{d_e} b_i x^i,$$ (3)

where $b_i$ denotes the polynomial coefficient. Moreover, let $N_e$ denote the root of $\Gamma_e$. MIE picks a random quantity $s = f_e(0) \in \mathbb{Z}_p$ and randomly chooses other $d_e$ points of $f_e(x)$ to determine $f_e(x)$. MIE selects $f_q(x)$ for the non-root node $N_q$ of $\Gamma_e$, where:
\[ f_q(0) = f_{parent(N_q)}(\text{index}(N_q)). \] \hspace{1cm} (4)

MIE randomly picks the other \( d_q \) points to determine \( f_q(x) \). Therefore, it can get the polynomials \( f_s(\cdot) \) for any \( N_q \in \Gamma_e \).

MIE chooses random values \( s_{1,0}, \ldots, s_{1,\omega_q-1}, \ldots, s_{k_e,0}, \ldots, s_{k_e,\omega_q-1} \in \mathbb{Z}_p \), and creates the mutual information \( MI \) as follows:

\[ \forall a_i \in \Gamma_e, i = 1, \ldots, k_e, \text{MIE first calculates:} \]

\[ C_{(i,0),1} = g^{s_{i,0}}, C_{(i,0),2} = (h_{i,0})^{s_{i,0}}. \]

Then, for \( j = 1 \) to \( \omega_i - 1 \), MIE calculates:

\[ C_{(i,j),1} = g^{s_{i,j}}, C_{(i,j),2} = (h_{i,j})^{s_{i,j}} \cdot H(a_i)^{s_{i,j-1}}. \]

When \( j = \omega_i \), let \( s_{i,j} = f_{a_i}(0) \), where \( f_{a_i}(\cdot) \) is the polynomial of the node whose attribute is \( a_i \). MIE calculates:

\[ C_{(i,j),1} = g^{f_{a_i}(0)}, C_{(i,j),2} = (h_{i,j})^{f_{a_i}(0)} \cdot H(a_i)^{f_{a_i}(0)}. \]

Then, MIE calculates:

\[ C_{(i,\text{end}),1} = g^{f_{a_i}(0)}, C_{(i,\text{end}),2} = (h_{i,\text{end}})^{f_{a_i}(0)}. \]

Finally, MIE creates the mutual information \( MI \):

\[ MI = \left\{ \Gamma_e, \{ C_{(i,j),1}, C_{(i,j),2} \}_{ \forall a_i \in \Gamma_e, j = 1, \ldots, \omega_i }, \{ C_{(i,\text{end}),1}, C_{(i,\text{end}),2} \}_{ \forall a_i \in \Gamma_e } \right\}. \]

Algorithm 1 shows the pseudocode of the \textbf{MutualInfoGen} algorithm.

\begin{algorithm}
\caption{The mutual information generation algorithm (MutualInfoGen).}
\textbf{Input:} \( PP \): The public parameter. \( \Gamma_1 \): The weighted access tree of Provider A. \( \Gamma_2 \): The weighted access tree of Provider B.
\textbf{Output:} \( MI \): The mutual information.
\begin{algorithmic}[1]
\State Create \( \Gamma_e \) according to the mutual sub-policy of \( \Gamma_1 \) and \( \Gamma_2 \).
\State Select a random \( s \in \mathbb{Z}_p \); set \( f_e(0) = s \); and randomly choose other \( d_e \) points to determine \( f_e(x) \).
\For {\( \forall \) non-root node \( N_q \in \Gamma_e \)}
\State Calculate \( f_q(0) \), and randomly choose other \( d_q \) points to determine \( f_q(x) \).
\If {\( a_i \in \Gamma_e \)}
\State Calculate \( C_{(i,j),1}, C_{(i,j),2} \).
\For {\( j = 1 \) to \( \omega_i \)}
\If {\( j \neq \omega_i \)}
\State Calculate \( C_{(i,j),1}, C_{(i,j),2} \).
\Else
\State Set \( s_{i,j} = f_{a_i}(0) \), and calculate \( C_{(i,j),1}, C_{(i,j),2} \).
\EndIf
\EndFor
\State Calculate \( C_{(i,\text{end}),1}, C_{(i,\text{end}),2} \).
\EndIf
\EndFor
\State \( MI = \left\{ \Gamma_e, \{ C_{(i,j),1}, C_{(i,j),2} \}_{ \forall a_i \in \Gamma_e, j = 1, \ldots, \omega_i }, \{ C_{(i,\text{end}),1}, C_{(i,\text{end}),2} \}_{ \forall a_i \in \Gamma_e } \right\} \).
\end{algorithmic}
\end{algorithm}

Encrypt\((PP, M_1, \Gamma_1, MI)\). MIE sends \( MI \) to the providers. Then, they encrypt the messages \( M_1, M_2 \), respectively. Consider Provider A as an instance. It runs Encrypt as follows. Provider A
creates the access tree $\Gamma_1$ based on $\Gamma_c$, which is a sub-tree of $\Gamma_1$. For any trunk node $N_c$ above $N_e$ in $\Gamma_1$, the polynomial $f_c(x)$ is determined as follows.

$$f_c(index(trunkchild(N_c))) = f_{trunkchild(N_c)}(0),$$

where $trunkchild(N_c)$ denotes the child node of $N_c$ along the trunk and $index(N_c)$ is the index value of $N_c$ to its parent node. In order to calculate conveniently, when providers construct the access policies, it sets $index(trunkchild(N_c)) = 1$ for each trunk node. For $d_c$ non-constant coefficients of (3), Provider A selects them randomly. Then, the constant term $f_c(0)$ can be computed by (10). Taking a simple instance, let $N_d$ be the parent of $N_c$. Provider A needs to determine:

$$f_d(x) = \sum_{i=0}^{d} b_i x^i.$$  

(11)

Provider A selects $d_d$ coefficients $\{b_1, b_2, \cdots, b_{d_d}\}$ randomly, and $index(trunkchild(N_d)) = 1$. Due to (10), Provider A knows that:

$$f_d(1) = f_c(0),$$

then it calculates the constant term $b_0$ with (3), as:

$$b_0 = f_c(0) - \sum_{i=1}^{d_d} b_i.$$  

(13)

Similarly, MIE determines every trunk node’s polynomial. For any other node $N_q$ of $\Gamma_1$, the method of determining $f_d(x)$ is identical to the method of MIE creating $\Gamma_c$.

Provider A obtains every polynomial of node in $\Gamma_1$ with (4) and (10), and $\Gamma_1$ is created completely. Provider B creates $\Gamma_2$ based on $\Gamma_c$ likewise.

Let $Y_1$ denote the set of the leaf nodes in $\Gamma_1$. The number of leaf nodes is denoted by $k_1$. Provider A produces the ciphertext $CT_1$ of message $M_1$ as follows:

Provider A first calculates $C_M = M_1 \cdot \overline{e(g, g)^{s_{k_1}(0)}}$, where $f_{N_1}(\cdot)$ is the polynomial of access tree $\Gamma_1$’s root node $N_1$. For $\forall a_i \in Y_1 \setminus Y_e, i = 1, 2, \cdots, k_1$, Provider A chooses random values $s_{k_1}, s_{k_1+1,0}, \cdots, s_{k_1+1,\omega_{k_1+1}} - 1, \cdots, s_{k_1,0}, \cdots, s_{k_1,\omega_{k_1} - 1} \in \mathbb{Z}_p$, and calculates:

$$C_{(i,0),1} = g^{s_{i,0}a_i}, C_{(i,0),2} = (h_{i,0})^{s_{i,0}}.$$  

(14)

Then, for $j = 1$ to $\omega_i - 1$, MIE calculates:

$$C_{(i,j),1} = g^{s_{i,j}} C_{(i,j),2} = (h_{i,j})^{s_{i,j}} \cdot \mathcal{H}(a_i)^{s_{i,j-1}}.$$  

(15)

When $j = \omega_i$, let $s_{ij} = f_{a_i}(0)$, where $f_{a_i}(\cdot)$ is the polynomial of the node whose attribute is $a_i$. MIE calculates:

$$C_{(i,j),1} = g^{f_{a_i}(0)} C_{(i,j),2} = (h_{i,j})^{f_{a_i}(0)} \cdot \mathcal{H}(a_i)^{s_{i,j-1}}.$$  

(16)

Then, MIE calculates:

$$C_{(i,end),1} = g^{f_{a_i}(0)} C_{(i,end),2} = (h_{i,end})^{f_{a_i}(0)}.$$  

(17)

Finally, Provider A creates the ciphertext $CT_1$:

$$CT_1 = \{ \Gamma_1, C_M, \{C_{(i,j),1}, C_{(i,j),2}\}_{Y_0 \in Y_1, j=1, \cdots, \omega_i}, \{C_{(i,end),1}, C_{(i,end),2}\}_{\forall a_i \in Y_1} \}.$$  

(18)

Algorithm 2 represents the pseudocode of **Encrypt** algorithm.
KeyGen(PP, MSK, S). WAA calls this algorithm to produce SK on the basis of MSK and the data consumer’s attributes. Particularly, \( \forall a_i \in S \), let \( \omega_i \) denote the weight of \( a_i \) and let \( (i, j) \) denote the \( j \)th part of \( a_i \), where \( j = 1, \cdots, \omega_i \). For each \( (i, j) \), WAA chooses \( D_{ij} \in G_1 \) and \( r_{ij} \in \mathbb{Z}_p \) randomly. WAA also chooses random values \( r_{i, \text{end}_i} \in \mathbb{Z}_p \). WAA generates SK components as follows:

\[
K_{(i,0),1} = D_{(i,0)}(h_{i,0})^{r_{i,0}}, K_{(i,0),2} = g^{r_{i,0}}, K_{(i,j),1} = D_{(i,j-1)}^{-1} \cdot H(a_i)^{r_{i,j}}/K_{(i,j),2} = g^{r_{i,j}},
\]

\[
K_{(i,j),3} = D_{(i,j)}(h_{i,j})^{r_{i,j}}, K_{(i,\text{end}_j),1} = g^{-a_i} \cdot D_{(i,j)}(h_{i,\text{end}_j})^{r_{i,\text{end}_j}}, K_{(i,\text{end}_j),2} = g^{r_{i,\text{end}_j}}.
\]

Algorithm 2 The encryption algorithm (Encrypt).

Input: PP: The public parameter. M1: The data that Provider A wants to share. \( \Gamma_1 \): The weighted access tree of Provider A. MI: The mutual information.

Output: CT1: The ciphertext of M1.

1: Create \( \Gamma_1 \) based on \( \Gamma_e \)
2: for \( \forall \) trunk node \( N_c \) above \( N_e \in \Gamma_1 \) do
3: Calculate \( f_c(1) \); select \( d_c \) non-constant coefficients randomly; and calculate \( f_c(0) \)
4: if \( N_c = N_1 \) then
5: Calculate \( C_M \)
6: end if
7: end for
8: for \( \forall \) non-trunk node \( N_0 \in \Gamma_1 \) do
9: Determine \( f_{i_0}(x) \) the same as MutualInfoGen
10: if \( a_i \in Y_1 \setminus Y_e \) then
11: Calculate \( C_{(i,j),1}, C_{(i,j),2} \)
12: for \( j = 1 \) to \( \omega_i \) do
13: if \( j \neq \omega_i \) then
14: Calculate \( C_{(i,j),1}, C_{(i,j),2} \)
15: else
16: Set \( s_{ij} = f_{a_i}(0) \), and calculate \( C_{(i,j),1}, C_{(i,j),2} \)
17: end if
18: end for
19: Calculate \( C_{(i,\text{end}),1}, C_{(i,\text{end}),2} \)
20: end if
21: end for
22: \( CT_1 = \{ \Gamma_1, C_M, \{C_{(i,j),1}, C_{(i,j),2}\}_{a_i \in Y_1, j=1, \cdots, \omega_i}, \{C_{(i,\text{end}),1}, C_{(i,\text{end}),2}\}_{a_i \in Y_1} \} \)

Then, WAA outputs:

\[
SK = \{ K_{(i,j),1}, K_{(i,j),2}r, K_{(i,j),1}, K_{(i,j),2}, K_{(i,j),3}, K_{(i,\text{end}),1}, K_{(i,\text{end}),2} \}_{a_i \in S, j=1, \cdots, \omega_i} ;
\]

Decrypt(PP, SK, CT1). Consider that the aforementioned consumer gets CT1, and he/she wants to recover M1. The decryption process is defined as follows.

1. If \( N_k \) is a leaf node of \( \Gamma_1 \) and is associated with attribute \( a_i \in S \) and \( \omega_i \leq \omega_i' \), let \( N_k \)'s Boolean value be TRUE, and calculate:
\[ F_{(x,0)} = \frac{\hat{e}(C_{(i,0),1}, K_{(i,0),1})}{\hat{e}(C_{(i,0),2}, K_{(i,0),2})} \\
= \frac{\hat{e}(g\text{\textsuperscript{x},0}, D_{(i,0)}(h_{1,i0})^\gamma_{1,i0})}{\hat{e}(h_{1,i0})^\gamma_{1,i0}, g^\gamma_{1,i0}} \\
= \frac{\hat{e}(g, D_{(i,0)})^{\gamma_{1,i0}} \cdot \hat{e}(g, h_{1,i0})^{\gamma_{1,i0} r_{1,0}}}{\hat{e}(g, h_{1,i0})^{\gamma_{1,i0} r_{1,0}}}
\]

(22)

Then, for \( j = 1 \) to \( \omega_i \):

\[ F_{(x,j)} = F_{(x,j-1)} \frac{\hat{e}(C_{(i,j-1),1}, K_{(i,j-1),1}) \cdot \hat{e}(C_{(i,j),1}, K_{(i,j),2})}{\hat{e}(C_{(i,j),2}, K_{(i,j),2})}
\]

(23)

When \( j = 1 \) to \( \omega_i - 1 \),

\[ F_{(x,j)} = \hat{e}(g, D_{(i,j)})^{\gamma_{1,i}}.
\]

(24)

When \( j = \omega_i \),

\[ F_{(x,j)} = \hat{e}(g, D_{(i,\omega_i)})^{f_{x}(0)}.
\]

(25)

Finally, the consumer computes:

\[ F_{x} = F_{(x,\omega_i)} \frac{\hat{e}(C_{(i,\text{end},2),1}, K_{(i,\text{end},2),1})}{\hat{e}(C_{(i,\text{end},1),1}, K_{(i,\text{end},1),1})} \\
= \frac{\hat{e}(g, D_{(i,\omega_i)})^{f_{x}(0)} \cdot \hat{e}(h_{1,\text{end}})^{f_{x}(0)} g^{r_{1,\text{end},1}}}{\hat{e}(h_{1,\text{end}})^{r_{1,\text{end},1}} D_{(i,\omega_i)} (h_{1,\text{end}})^{r_{1,\text{end},1}}}
\]

(26)

\[ a_i \] is the attribute of leaf node \( N_x \), so \( f_{x_i}(\cdot) = f_{x}(\cdot) \), and:

\[ F_{x} = \hat{e}(g, g)^{a f_{x}(0)} = \hat{e}(g, g)^{a f_{x}(0)}.
\]

(27)

Equations (22) and (24)–(26) are on the basis of bilinear mapping properties, which were introduced in Section 3.1. If \( a_i \in S, \omega_i > \omega_i ' \) or \( a_i \not\in S \), then \( F_x = \bot \), and \( \bot \) is a termination signal.

For a non-leaf node \( N_x \) of \( \Gamma_1 \), let \( N_x \) be the children of \( N_x \) and \( S_x \) be an arbitrary \( V_x \)-size set of \( N_x \), where \( F_x \neq \bot \). If \( S_x \) does not exist, set \( F_x = \bot \), or else, \( N_x \)'s Boolean value is TRUE, then compute:

\[ F_{x} = \prod_{z \in S_x} F_{z}^{\Delta_{z,S_x}(0)} \\
= \prod_{z \in S_x} \hat{e}(g, g)^{a f_{z}(0) \Delta_{z,S_x}(0)} \\
= \prod_{z \in S_x} \hat{e}(g, g)^{a f_{z}(z) \Delta_{z,S_x}(0)} 
\]

(28)

where \( \Delta_{z,S_x}(y) = \prod_{i \in S,z \neq j} \frac{y}{y - j} \) is the Lagrange coefficient polynomial.

Therefore, if the access tree \( \Gamma_1 \) is satisfied by \( S \), the decryption algorithm begins from the root node \( N_1 \), and the consumer calculates:
Algorithm 3 displays the pseudocode of the Decrypt algorithm. Figure 4 shows the working process of UCP-WABE for unified encryption. Figure 5 displays the process flow of a consumer for decryption.

Algorithm 3 The decryption algorithm (Decrypt).

Output: M1: The plaintext of CT1.

1: for ∀ leaf node N_x ∈ Γ_1 do
2: if attribute of N_x ∈ S, and ω_i ≤ ω'_i then
3: Set N_x’s Boolean value to be TRUE, and calculate F_{(x,0)}
4: for j = 1 to ω_i do
5: if j ≤ ω_i − 1 then
6: Calculate F_{(x,j)} = \hat{e}(g, D(i,j))^{s_i,j}
7: else
8: Calculate F_{(x,j)} = \hat{e}(g, D(i,\omega_j))^{f_i(0)}
9: end if
10: end for
11: Calculate F_x
12: else
13: Set F_x = ⊥
14: end if
15: end for
16: for ∀ non-leaf node N_x ∈ Γ_1 do
17: Set N_z as the children of N_x and S_x as an arbitrary V_x-size set of N_z, where F_z ≠ ⊥.
18: if ∄ S_x then
19: Set F_x = ⊥
20: else
21: Set N_x’s Boolean value to be TRUE, and calculate F_x
22: end if
23: end for
24: if F_{N_1}'s Boolean value is TRUE then
25: Calculate \frac{C_M}{F_{N_1}}
26: end if
27: M_1 = \frac{C_M}{F_{N_1}}

Figure 4. The process flow of unified encryption.
6. Security Analysis

6.1. Security Proof

Extended from CP-WABE, UCP-WABE is expected to have the same security property as CP-WABE, which has been proven to be selectively secure under a variant of the decision ℓ-bilinear Diffie-Hellmann exponent (BDHE) assumption [16]. Based on the selective security of CP-WABE, we prove UCP-WABE is selectively secure.

Theorem 1. If no polynomial time adversary can selectively break CP-WABE with a weighted challenge access policy Γ∗, no polynomial time adversary can selectively break UCP-WABE with Γ∗.

Proof. To prove the theorem, we assume that there exists a polynomial time adversary A, which has a non-negligible advantage AdvA in selectively breaking UCP-WABE. Using A, we will build a polynomial time adversary B, which selectively breaks CP-WABE with a non-negligible advantage AdvB.

Init: A declares a challenge weighted access policy Γ∗ and sends it to B. B sends Γ∗ to the CP-WABE challenger.

Setup: The public parameter of CP-WABE PP = \( \left\{ G_1, g, \tilde{e}(g, g)^a, \{ h_{i,\text{end}}, h_i \} \right\}_{i \in \mathbb{U}, j = 0, \ldots, \ell} \) is sent to B, then B sends PP′ = PP to A.

Phase 1: A makes secret key queries to B for attribute sets, and none of these sets satisfy the challenge access policy Γ∗. Suppose A asks for attribute set S_{q1}. B gives S_{q1} to the CP-WABE challenger and gets SK_{q1} = \( \left\{ K_{(i,0),1}, K_{(i,0),2}, K_{(i,j),1}, K_{(i,j),2}, K_{(i,\text{end})}, 1, K_{(i,\text{end})}, 2 \right\}_{i \in \mathbb{U}, j = 1, \ldots, \omega} \). B returns SK′_{q1} = SK_{q1} to A to answer the query. This process would repeat until A no longer queries.

Challenge: A submits two messages of the same length m_0, m_1. Then, A constructs a weighted access policy Γ′ that there is a mutual access sub-policy between Γ∗ and Γ′. Γ′ is also sent to B. B sends m_0, m_1 to the CP-WABE challenger. The CP-WABE challenger flips a random coin b \( \leftarrow \mathbb{R} \) \{0, 1\} and encrypts m_b with Γ′. The ciphertext is CT = \( \left\{ \Gamma, C_{m_b}, C_{(i,j),1}, C_{(i,j),2} \right\}_{i \in \mathbb{U}, j = 1, \ldots, \omega} \), \( \left\{ C_{(i,\text{end})}, 1, C_{(i,\text{end})}, 2 \right\}_{i \in \mathbb{Y}} \), and the CP-WABE challenger sends it to B. B constructs challenge ciphertext CT∗ as CT∗ = CT and returns it to A.

Phase 2: A asks for more secret keys, and the same restriction is that Γ′ must not be satisfied by any one of the queried attribute sets. B answers as in Phase 1.

Guess: A outputs its guess b′ \( \in \{0, 1\} \), and B outputs b′ in its own game. B has an advantage in breaking CP-WABE as

\[
Adv_B = |Pr[b = b'] - 1/2| = Adv_A
\]

Thus, B has a non-negligible advantage in selectively breaking CP-WABE, so the proof of the theorem is completed. □
6.2. Discussion

In UCP-WABE, CSP cannot know the details of the data because the data are stored in an encrypted form. The data are encrypted under a weighted access policy, so only the consumers who possess a set of weighted attributes satisfying the policy can decrypt the ciphertext. Hence, UCP-WABE achieves a fine-grained access control, and the confidentiality of the data can be guaranteed against unauthorized consumers. A data provider constructs the weighted access policy according to the data consumers’ attributes instead of the data consumers’ detailed information. The threshold gates are also used to construct the weighted access policy. Therefore, the providers are able to construct sophisticated weighted access policies, and there is no limit to the number of threshold gates and attributes. Thus, UCP-WABE can support an extremely large number of weighted access policies, and UCP-WABE can implement flexible access control.

7. Evaluation of the Encryption Efficiency

The theoretical efficiency analysis of UCP-WABE is first given, then we carry out the experiments, and the results show that UCP-WABE has better efficiency than other schemes.

7.1. Efficiency Analysis Based on Encryption Methodologies

Let \( \Gamma \) denote the least node set that satisfies \( \Gamma_1 \). In order to analyze the efficiency conveniently, suppose the number of levels of \( \Gamma_1 \) is equal to \( \Gamma_2 \)’s and each level has the same number of attributes. Thus, \( \Gamma_1, \Gamma_2 \) have the same number of attributes, i.e., \( k_1 = k_2 \). Let \( B(*) \) be the bit size of an element of \(*\) and \( |*| \) be the number of elements of \(*\). Let \( S(\Gamma_1) \) denote the least node set that satisfies \( \Gamma_1 \). The comprehensive comparison among CP-ABE [24], CP-WABE [16], and our proposed UCP-WABE is summarized in Table 2. Moreover, the hash computation cost does not have to be included because it is very small.

<table>
<thead>
<tr>
<th>Component</th>
<th>CP-ABE</th>
<th>CP-WABE</th>
<th>UCP-WABE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public key size</td>
<td>( 3B(G_1) + B(G_2) )</td>
<td>( (\sum_{i=1}^{l} l_i + 2</td>
<td>\mathcal{U}</td>
</tr>
<tr>
<td>Master key size</td>
<td>( B(Z_p) + B(G_1) )</td>
<td>( B(G_1) )</td>
<td>( B(G_1) )</td>
</tr>
<tr>
<td>Secret key size</td>
<td>( (2</td>
<td>\mathcal{S}</td>
<td>+ 1)B(G_1) )</td>
</tr>
<tr>
<td>Ciphertext size</td>
<td>( \sum_{i=1}^{k_1} (l_i - \omega_i + 1) \cdot )</td>
<td>( 4\sum_{i=1}^{l_1} \omega_i + 1)B(G_1) + 2B(G_2) )</td>
<td>( 4\sum_{i=1}^{l_1} \omega_i + 1)B(G_1) + 2B(G_2) )</td>
</tr>
<tr>
<td></td>
<td>( \left[ (4k_1 + 2)B(G_1) + 2B(G_2) \right] )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encryption Time</td>
<td>( \frac{1}{k_1} \sum_{i=1}^{k_1} (l_i - \omega_i + 1) \cdot )</td>
<td>( 8\sum_{i=1}^{l_1} \omega_i + 8k_1)C(G_1) + 4C(G_2) )</td>
<td>( 8\sum_{i=1}^{l_1} \omega_i + 8k_1 - (4\sum_{i=1}^{l_1} \omega_i + 4k_1)C(G_1) + 4C(G_2) )</td>
</tr>
<tr>
<td></td>
<td>( \left[ (4k_1 + 2)C(G_1) + 4C(G_2) \right] )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decryption Time</td>
<td>( \frac{2(</td>
<td>\mathcal{S}</td>
<td>+ 1)}{2(</td>
</tr>
<tr>
<td>Weighted Attributes</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Unified Encryption</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

As shown in Table 2, suppose the values of \( l_i \) are given. When \( k_1 \) and \( \omega_i \) are fixed, the encryption time cost declines with \( k_\varphi \) in UCP-WABE. In order to display the results more intuitively, we suppose all the attributes have the same weight in the access trees \( \Gamma_1, \Gamma_2 \) and all the maximum weights are the same in \( \mathcal{U} \). Therefore, the time cost of encryption linearly declines with \( k_\varphi \), and the rate of
descent is \(4(\omega_i + 1)C(G_1)\) in UCP-WABE. In CP-ABE and CP-WABE, the encryption time cost remains unchanged. When \(k_e\) and \(\omega_i\) are fixed, the encryption time cost linearly rises with \(k_1\), and the growth rate is \(8(\omega_i + 1)C(G_1)\) in UCP-WABE. Although the growth rate of encryption time cost is also equal to \(8(\omega_i + 1)C(G_1)\) in CP-WABE, it is always higher than that in UCP-WABE. The encryption time cost increases with \(k_1\) exponentially in CP-ABE. When \(k_1\) and \(k_e\) are fixed, the encryption time cost linearly rises with \(\omega_i\), and the growth rate is \((8k_1 - 4k_e)C(G_1)\) in UCP-WABE. The encryption time cost also linearly rises with \(\omega_i\), and the growth rate is \(8k_1 C(G_1)\) in CP-WABE. In spite of the encryption time cost declining with \(\omega_i\) in CP-ABE, it still remains far higher than UCP-WABE. In Table 2, one noteworthy fact is that the encryption time cost is a summation of the two data providers’ encryption time.

7.2. Efficiency Analysis Based on Implementation

The implementation of UCP-WABE is on the basis of the cpabe toolkit [33] and the Pairing-Based Cryptography library [34]. The implementation uses a 160-bit elliptic curve group based on the supersingular curve \(y^2 = x^3 + x\) over a 512-bit finite field. The experiments are conducted on a PC, in Intel Core2 Duo with 3.00-GHz CPU and 2GB RAM, running Ubuntu15.04. We compare UCP-WABE with CP-WABE and CP-ABE in the field of encryption time cost. For all experiments, we make all the weighted access policies’ threshold gates “AND” gates. This ensures that all of the ciphertext components could be calculated in Decrypt.

Figure 6 shows the encryption time cost with given experimental conditions \(k_1 = 27, \omega_i = 3, l_i = 4\) \((i = 1, \cdots, 7)\), \(l_j = 3\) \((i = 8, \cdots, 27)\). We do not suppose all the attributes have the same maximum weight in this case for the reason that the result of CP-ABE is too high. The number of mutual attributes used in the experiments is \(k_e = \{2, 5, 8, 11, 14, 17, 20, 23, 26\}\). As shown in Figure 6, the encryption time cost of UCP-WABE follows a linear decline with the number of the mutual access tree attributes. For CP-WABE and CP-ABE, the encryption time cost almost keeps unchanged.

Figure 6. Encryption time cost for the changed number of mutual access tree attributes.

Figure 7 displays the encryption time with \(k_e = 3, \omega_i = 3\) and \(l_i = 4\). In the experiments, the number of \(\Gamma_1’s\) attributes is \(k_1 = \{4, 5, 6, 7, 8, 9, 10\}\). As shown in Figure 7, the encryption time cost follows a linear growth with the number of the access tree attributes in UCP-WABE and CP-WABE. Meanwhile, the former is lower than the latter. The result of CP-ABE increases exponentially with the number of access tree attributes.
Figure 7. Encryption time cost for the changed number of access tree attributes.

Figure 8 shows the encryption time cost with given experimental conditions $k_1 = 3, k_e = 2$, and $l_i = 9$. The weight of attributes in $\Gamma_1$ used in the experiments is $\omega_i = \{1, 2, 3, 4, 5, 6, 7\}$. Figure 8 shows that the encryption time cost of UCP-WABE and CP-WABE is following a linear growth in the weight of attributes, and the former is lower than the latter. The result of CP-ABE is decreasing with the weight of attributes; however, it is still higher than UCP-WABE. Therefore, UCP-WABE improves the efficiency of encryption.

Figure 8. Encryption time cost for the changed weight of attributes in the access tree.

8. Conclusions

In this paper, we have proposed a novel UCP-WABE scheme for sharing data in cloud computing environments. The scheme optimizes the encryption of data that are encrypted under the multi-providers’ access policies where these policies have a mutual sub-policy. UCP-WABE utilizes a mutual information extractor to extract the mutual sub-policy and produce mutual information, which assists in optimizing the encryption. UCP-WABE also takes advantage of weighted attribute-based encryption to avoid too complex of an access structure. We analyse the computational complexity of UCP-WABE theoretically and experimentally. The analyses indicate that UCP-WABE has a better efficiency of encryption. The security analysis shows that UCP-WABE is selectively secure. It should be noted that we only consider the case of two data providers. In practice, UCP-WABE will
have a more efficient data sharing if the more providers’ access policies have a mutual sub-policy. In our future work, we will optimize the system implementation and conduct comprehensive experiments with real-life cases in cloud computing.

**Author Contributions:** W.L. proposed the main idea and conceptualization. He also performed experiments, results analysis, and scientific discussions and wrote the paper. W.N., D.L. and R.P.L. helped to revise the clarity of the work, as well as write and organize the paper. Finally, S.L. assisted in English corrections and submission of the article.

**Funding:** This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802300, in part by the National High-tech R&D Program of China (863 Program) under Grant 2015AA016005 and Grant 2015AA017201, in part by the Applied Sci-Tech R&D Special Fund Program of Guangdong Province under Grant 2015B010131007 and in part by the China Scholarship Council under Grant 201506470040.

**Acknowledgments:** The authors thank the reviewers for their valuable comments and suggestions, which improved the technical content and the presentation of the paper.

**Conflicts of Interest:** The authors declare that they have no conflict of interest.

**Abbreviations and Notations**

The following abbreviations and notations are used in this manuscript:

- **UCP-WABE** unified ciphertext-policy weighted attribute-based encryption
- **M**\(_i\) the data that provider \(i\) wants to share
- **\(\Gamma_i\)** the weighted access tree relates to \(M_i\)
- **\(\tilde{e}(\cdot)\)** a bilinear mapping
- **\(\mathcal{G}_1\)** \(\tilde{e}(\cdot)\)'s input group
- **\(\mathcal{G}_2\)** \(\tilde{e}(\cdot)\)'s output group
- **\(p\)** \(\mathcal{G}_1\)'s order, namely there are \(p\) elements in \(\mathcal{G}_1\)
- **\(\mathbb{Z}_p\)** \(\mathcal{G}_1\)'s set of integers \([0, p−1]\)
- **\(g\)** \(\mathcal{G}_1\)'s generator. \(\forall i \in \mathbb{Z}_p, g^i \in \mathcal{G}_1\)
- **\(N_j\)** the \(j\)th node in \(\Gamma\)
- **\(\text{num}_j\)** \(N_j\)'s children number
- **\(V_j\)** \(N_j\)'s threshold value
- **\(S\)** the attribute set of a data consumer
- **\(a_i\)** the \(i\)th attribute of a data consumer
- **WAA** weighted attribute authority
- **MIE** mutual information extractor
- **\(PP, MSK\)** the public parameter and master secret key of UCP-WABE
- **SK** secret key of a data consumer, issued by WAA
- **\(U\)** the set of all attributes in the system
- **\(u_i\)** the \(i\)th attribute in \(U\)
- **\(l_i\)** the maximum weight of \(u_i\)
- **\(\omega_i\)** the weight of \(a_i\) from \(\Gamma\)
- **\(\omega_i'\)** the weight of \(a_i\) from \(S\)
- **\(k_i\)** the number of leaf nodes in \(\Gamma_i\)
- **\(N_{e}\)** the root node of the mutual access tree
- **\(MI\)** the mutual information
- **\(H(\cdot)\)** the hash function to hash an attribute
- **\(CT_i\)** the ciphertext of \(M_i\)
- **\(\perp\)** termination signal
- **\(C(\cdot)\)** the operation in \(\mathcal{G}_i\)
- **\(C_{\tilde{e}}\)** the operation in \(\tilde{e}(\cdot)\)
- **\(B(\ast)\)** the bit size of an element of \(\ast\)
- **\(|\ast|\)** the number of elements of \(\ast\)
References


© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).