Arbitrary-State Attribute-Based Encryption with Dynamic Membership

Chun-I Fan, *Member*, *IEEE*, Vincent Shi-Ming Huang, and He-Ming Ruan

Abstract—Attribute-based encryption (ABE) is an advanced encryption technology where the privacy of receivers is protected by a set of attributes. An encryptor can ensure that only the receivers who match the restrictions on predefined attribute values associated with the ciphertext can decrypt the ciphertext. However, maintaining the correctness of all users' attributes will take huge cost because it is necessary to renew the users' private keys whenever a user joins, leaves the group, or updates the value of any of her/his attributes. Since user joining, leaving, and attribute updating may occur frequently in real situations, membership management will become a quite important issue in an ABE system. In this paper, we will present an ABE scheme which is the first ABE scheme that aims at dynamic membership management with arbitrary states, not binary states only, for every attribute. Our work also keeps high flexibility of the constraints on attributes and makes users be able to dynamically join, leave, and update their attributes. It is unnecessary for those users who do not change their attribute statuses to renew their private keys when some user updates the values of her/his attributes. Finally, we also formally prove the security of the proposed scheme without using random oracles.

Index Terms—Arbitrary-state attributes, dynamic membership, attribute-based encryption (ABE), bilinear pairing

1 Introduction

In 1993, Fiat and Naor proposed a broadcast encryption system [1] which is useful in many commercial applications such as pay-TV. However, in some applications, we may want to broadcast an encrypted message to a specific sub-group but not all of the users. The broadcast encryption systems cannot fill such requirement. The notion of attribute-based encryption (ABE) was first proposed in 2005 [2], which makes it possible for an encryptor to select a set of attributes such that only the receivers with these attributes can decrypt the ciphertext.

In an ABE system, an encryptor does not know who can decrypt the ciphertext and, meanwhile, the receivers cannot know who the encryptor is. This provides privacy protection for the encryptor and receivers such that they can exchange secret information and gain anonymity simultaneously. In many situations, one may want to encrypt some sensitive data which can only be decrypted by the users who meet the designated constraints. For example, someone named Bernie Rhodenbarr wants to share his experience of struggling against mental disorder, say melancholia, with other male patients who have the same problem on a platform such as YouTube or any forum. However, the normal people do not understand those mental disorders and may have stereotypes against those patients. Therefore, Bernie will hope that only those who have the same problem as him or are professional doctors at such disease can read the message. Thus Bernie can designate the following constraint for accessing the message such as ((“GENDER: MALE” AND “MENTAL DISORDER: MELANCHOLIA”) OR (“CAREER: DOCTOR” AND “SPECIALITY: MELANCHOLIA”)).

From 2005 to 2009, many schemes [3]–[15] have been proposed to solve the above problem. However, the constraints on attributes provided by the above schemes are just in the format of *satisfying* or *not satisfying* some attributes, i.e., the attribute value only has a binary state, “0” or “1”, which cannot deal with arbitrary-state attributes, such as “CAREER: DOCTOR, PROFESSOR, ENGINEER”. Our proposed scheme will extend binary-state attributes to arbitrary-state ones such that it becomes more practical for real applications. Our proposed scheme is the first one which can support arbitrary-state attributes. Owing to the feature, our scheme can also be used as an identity-based encryption scheme [16] because the identity attribute is arbitrary-state, too.

Besides, [4]–[6] mentioned that some users may be compromised and thus, for some applications, like pay-TV, the service provider will suffer commercial damage in such circumstance. Hence, [4]–[6] aimed at providing a revocation mechanism to deny the accesses of the compromised users. The revoked users cannot decrypt new ciphertexts anymore even though their attributes satisfy the constraints of the ciphertexts.

In our scheme, we provide a dynamic membership mechanism which covers not only user revocation but also user enrollment and attribute updating. Dynamic membership is always required in most applications. For example, in a pay-TV system, the attributes of a user are defined as the channels she/he orders. In the system, a user may want to order new channels or cancel some ordered channels after using the

- *C.-I. Fan is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan. E-mail: cifan@faculty.nsysu.edu.tw.*
- *V.S.-M. Huang is with the Cloud Computing Center for Mobile Application, Industry Technology Research Institute, Hsinchu 30011, Taiwan. E-mail: Vincent.SM.Huang@gmail.com.*
- *H.-M. Ruan is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan. E-mail: tannhauser.sphinx@gmail.com.*

Manuscript received 11 Aug. 2012; revised 03 Feb. 2013; accepted 26 Mar. 2013. Date of publication 07 Apr. 2013; date of current version 15 July 2014. Recommended for acceptance by J.L. Beuchat.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2013.83

system for a period of time. The attribute updating property can help the system to achieve this goal. In addition, users may want to quit the system and new users may join the service anytime. Therefore, it is necessary to consider member revocation and enrollment in the system. In order to make ABE more practical, dynamic membership is also desired.

1.1 Related Works

In 2001, Boneh and Franklin proposed an identity-based encryption (IBE) scheme based on Weil pairing [16], and enhanced it by the technique from Fujisaki and Okamoto [17]. In 2005, Sahai and Waters proposed a new type of IBE [2], [11], fuzzy IBE, which was the prototype of ABE, but it allows only fixed identities in the system and seems much close to an access control mechanism. In 2007, Baek, Susilo, and Zhou presented another fuzzy IBE with new construction [3]. This can be regarded as a much more mature ABE scheme, but it supports constraints only t-out-of-n threshold, that is, one can decrypt the ciphertext if and only if she/he meets not fewer than t in the n requirements constrained by the encryptor.

From 2007, more ABE schemes have been proposed. Most of these schemes possess specific features. For example, [7] provided full logic expression, (AND, OR, NOT), for attribute constraints. [8] supported key delegation which allows a user to issue a private key with a subset of the attributes owned by herself/himself. [9] provided threshold multiple authorities where attribute keys are issued by multiple authorities and a user can decrypt a ciphertext only if she/he has at least d_k attribute keys given from each authority k. [12] is a dual-policy ABE system which conjunctively combined Key-Policy and Ciphertext-Policy solutions. [13] and [15] provided distributed attribute authorities and privacy preservation, respectively. However, only [4]–[6] aimed at member management. The member management mechanism of [4]–[6] is designed to revoke the compromised users. They do not possess the feature of dynamic membership.

1.2 Our Contributions

Our scheme is the first one which supports dynamic membership and arbitrary-state attributes. Users can dynamically join and leave from our ABE system. This makes the ABE system more practical for real situations. Users can also change the values of their attributes which were given when they joined the system. Besides, the domain of each attribute is not just a binary bit but a variable-length string, and thus it can support arbitrary-state attributes. Furthermore, users can modify their attribute values when their statuses are changed. It is unnecessary for those users who do not change their attribute statuses to communicate with Key Generation Center (KGC) and refresh their private keys. Finally, we formally prove the security of the proposed ABE scheme under a chosen-ciphertext attack (CCA) model without using random oracles.

Note that the combination of binary-state attributes is not an appropriate solution for construction of an arbitrary-state attribute. This is because that the system should predict and prepare all possible public parameters and values for these binary-state attributes in advance and meanwhile the system parameters will become very complex. In our scheme, the set of the possible contents of an arbitrary-state attribute can be defined until encryption, and it can be re-defined whenever encryption, too. Thus, it is unnecessary for the system to predefine the contents of the attributes to cover all possible use-cases. Hence, the combination of binary-state attributes will not achieve the goal that an arbitrary-state attribute can achieve.

2 Preliminaries

2.1 Background

2.1.1 *Lagrange Interpolation*

According to [18], [19], a Lagrange interpolating polynomial is a polynomial p of degree not greater than $(n-1)$ that passes through n points $(x_i, y_1), \ldots, (x_n, y_n)$, and is given by

$$p(x) = \sum_{j=1}^{n} p_j(x), \text{where } p_j(x) = y_j \prod_{k=i,\ldots,n,k\neq j} \frac{x - x_k}{x_j - x_k}.$$

For $i \in \mathbb{Z}$ and $S \subseteq \mathbb{Z}$, the Lagrange coefficient $\Delta_{i,S}(x)$ is defined as

$$\Delta_{i,S}(x) = \prod_{\forall j \in S, j \neq i} \frac{x - j}{i - j}.$$

2.1.2 *Bilinear Mapping*

Let G_0, G_1, and G_T be three cyclic groups of prime order q. A bilinear mapping $e : G_0 \times G_1 \rightarrow G_T$ satisfies the following properties [16]:

- **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$, $\forall P \in G_0$, $Q \in G_1$ and $a, b \in \mathbb{Z}_q$.
- **Non-Degeneracy:** The mapping does not map all pairs in $G_0 \times G_1$ to the identity in G_T. Since G_0, G_1 are groups of prime order, this implies that if P and Q are generators of G_0 and G_1, respectively, then $e(P, Q)$ is a generator of G_T.
- **Computability:** There is an efficient algorithm to compute $e(P, Q), \forall P \in G_0, Q \in G_1$.

2.1.3 *Key Policy vs. Ciphertext Policy*

From [8], we can classify ABE into the following two types according to the relationship between access rules and private keys.

- **Key-Policy Schemes:** An access rule is issued to a user with her/his private keys at the private key generation phase. Thus the user can just decrypt the ciphertexts under the given access rule.
- **Ciphertext-Policy Schemes:** An access rule belongs to a ciphertext and is decided when the ciphertext is created. The access rule is independent of users' private keys. Thus a user can decrypt ciphertexts under different access rules as long as her/his attributes satisfy the rules.

It is clear that the ciphertext-policy schemes gain much more flexibility and are more suitable for general purposes.

2.1.4 *The Decisional Bilinear Diffie-Hellman (DBDH) Problem*

Let G_0, G_1, and G_T be three cyclic groups of prime order q, P and Q be arbitrarily-chosen generators of G_0 and G_1, respectively, and $e : G_0 \times G_1 \rightarrow G_T$ be a bilinear mapping. Given $(P, Q, aP, bP, cP, aQ, bQ, cQ, Z)$ for some $a, b, c \in \mathbb{Z}_q^*$ and $Z \in G_T$, decide if $Z = e(P, Q)^{abc}$.

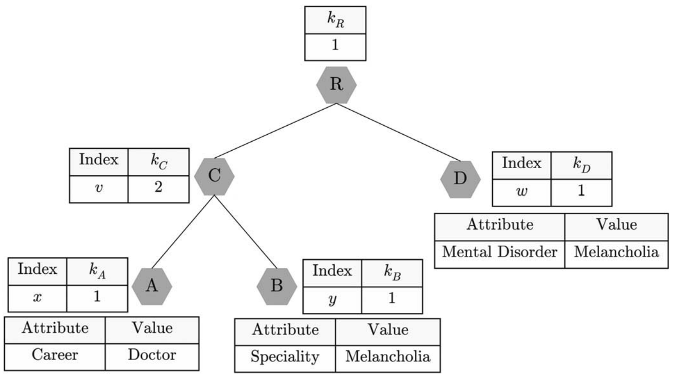

Fig. 1. An example of access tree structure.

2.2 Definitions

Definition 1. (The DBDH Assumption [20]). *We define that an adversary $\mathcal{C}$ with an output $b' \in \{0,1\}$ has advantage ϵ' in solving the DBDH problem if*

$$|Pr[\mathcal{C}(P,Q,aP,bP,cP,aQ,bQ,cQ,e(P,Q)^{abc})=1]- \\ Pr[\mathcal{C}(P,Q,aP,bP,cP,aQ,bQ,cQ,Z)=1]| \geq \epsilon',$$

where the probability is over the random choice of $a,b,c \in \mathbb{Z}_q^*$ and the random choice $Z \in G_T$. We say that the DBDH assumption holds if no polynomial-time adversary has non-negligible advantage in solving the DBDH problem.

Definition 2 (Access Tree Structure $\mathcal{T}$). *An access tree structure $\mathcal{T}$ is used to represent a logic expression of the given attributes (leaves) in our proposed scheme. $\mathcal{T}$ contains (T_S, $\mathcal{N}$, $\mathcal{I}$, $\mathcal{K}$, $A_\mathcal{T}$, $(m_j)_{\forall j \in A_\mathcal{T}}$) which are described as follows.*

- *T_S: a tree diagram.*
- *$\mathcal{N}$: the vector of all nodes in T_S. For example, $\mathcal{N}=(A,B,C,D,R)$ in Fig. 1.*
- *$\mathcal{I}$: the vector of the indexes of all nodes except the root node in T_S. For example, $\mathcal{I}=(x,y,v,w)$ in Fig. 1.*
- *$\mathcal{K}$: the vector of the degrees of the polynomials associated with the nodes in T_S, such as $\mathcal{K}=(i,1,2,1,1)$ in Fig. 1.*
- *$A_\mathcal{T}$: a set containing the attributes of all leaf nodes in T_S, for example, $A_\mathcal{T}=\{$'Career', 'Speciality', 'Mental Disorder'$\}$ in Fig. 1.*
- *$(m_j)_{\forall j \in A_\mathcal{T}}$: the vector of the values of the attributes in $A_\mathcal{T}$. For example, $(m_j)_{\forall j \in A_\mathcal{T}}=$ ('Doctor', 'Melancholia', 'Melancholia') in Fig. 1.*
- *Access rule: According to $\mathcal{K}$, the access tree structure $\mathcal{T}$ implies an access rule. For example, the access rule deduced from Fig. 1 is ("Career: Doctor" AND "Speciality: Melancholia") OR ("Mental Disorder: Melancholia").*
- *Satisfying $\mathcal{T}$: To satisfy an access tree structure $\mathcal{T}$, one has to possess the attributes A with attribute values $(m_j)_{\forall j \in A}$ where $(A,(m_j)_{\forall j \in A})$ satisfy the access rule deduced from $\mathcal{T}$. For example, one possesses attribute "Mental Disorder" with value "Melancholia", and thus she/he can satisfy the access rule in Fig. 1; another one who possesses attributes ("Career", "Speciality") with values ("Doctor", "Melancholia") can also satisfy the access rule.*

A ciphertext-policy attribute-based encryption scheme with dynamic membership (CP-ABE-DM) is defined as follows.

Definition 3 (CP-ABE-DM). *CP-ABE-DM contains the following algorithms.*

- ***Setup**$(k) \rightarrow (PK, MK)$: Take a security parameter k as input and generate public parameter PK and master key MK.*
- ***Enrollment**$(S_i) \rightarrow \mathcal{D}_i$: The input is $S_i=(A_i,(m_{i,j})_{\forall j \in A_i})$ where A_i is the set of user i's attributes and $(m_{i,j})_{\forall j \in A_i}$ is the vector of the values corresponding to the attributes in A_i. This algorithm will be performed when user i enrolls in the system. The private key $\mathcal{D}_i$ will be issued to user i.*
- ***Leaving**(i): When user i is revoked by the system, this algorithm will be performed and then PK will be updated.*
- ***Updating**$(i,j,m) \rightarrow \mathcal{D}_i$: This algorithm will be executed when user i applies for modifying her/his attribute j to be with value m. After performing this algorithm, the updated $\mathcal{D}_i$, will be given to user i and PK will also be updated.*
- ***Encryption**$(PK,\mathcal{T},K,M) \rightarrow CT$: The inputs are the public parameter PK, an access tree structure $\mathcal{T}$, a key K, and a message M. This algorithm computes $E_K(M)$ and encrypts K under $\mathcal{T}$ where E_K is a symmetric encryption algorithm with input key K. Then it outputs a ciphertext CT.*
- ***Decryption**$(CT,\mathcal{D}_i) \rightarrow M$: This algorithm takes a ciphertext CT and the private key $\mathcal{D}_i$ of user i as input. If user i's $(A_i,(m_{i,j})_{\forall j \in A_i})$ satisfy the access tree structure in CT, the algorithm will decrypt the ciphertext and output the message M; otherwise $\perp$ will be returned.*

Definition 4. (Dynamic Membership). *A ciphertext-policy attribute-based encryption scheme with dynamic membership should satisfy the following properties:*

- ***Expandability.** A new user is able to enroll in the system for the ABE service.*
- ***Renewability.** Each user's private key, attribute set, and attribute values can be renewed and the old private keys should be useless to those ciphertexts which are encrypted after these parameters associated with the user are renewed.*
- ***Revocability.** A user's private key can be revoked and the revoked private keys should be useless to those ciphertexts which are encrypted after these private keys are revoked.*
- ***Independence.** When a user's leaving or attribute updating occurs, the other users in the system are not required to interact with KGC (Key Generation Center) to renew their private keys.*

Definition 5. (The CCA_{DM} Game). *Let $\mathcal{A}$ be an adversary and Π be a ciphertext-policy ABE scheme with dynamic membership. $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ in the following game:*

- ***Setup:** $\mathcal{C}$ runs the Setup algorithm and gives the public parameter, PK, to $\mathcal{A}$.*
- ***Phase 1:** $\mathcal{A}$ can make the following queries:*
  - ***Enrollment**(S_i): This query is used to extract user i's private key $\mathcal{D}_i$ corresponding to $S_i=(A_i,(m_{i,j})_{\forall j \in A_i})$ where A_i is the set of the attributes of user i and $(m_{i,j})_{\forall j \in A_i}$ are the values of the attributes, respectively, in A_i. Finally, $\mathcal{D}_i$ will be recorded in a set $\mathcal{L}_D$.*
  - ***Leaving**(i): This query is used to revoke the private key of user i. After revoking, PK is updated and $\mathcal{D}_i$ is removed from $\mathcal{L}_D$.*
  - ***Updating**(i,j,m): This query is used to update the value of attribute j of user i to be m. After updating, the updated private key will be given to $\mathcal{A}$ and both PK and $\mathcal{L}_D$ will also be updated accordingly.*

– ***Decryption***$'(CT)$: *This query is used to decrypt a given ciphertext CT and output the message M corresponding to CT. If CT is not a correctly-constructed ciphertext, it outputs $\perp$.*

- ***Challenge:*** *At this phase, $\mathcal{A}$ can decide to perform **Challenge 1** or **Challenge 2**.*
  – ***Challenge 1:*** *$\mathcal{A}$ submits $(M_0, M_1, \mathcal{T}^*)$ to $\mathcal{C}$ where M_0 and M_1 are two equal-length messages and $\mathcal{T}^*$ is an access tree structure. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$ and $K \in G_T$. Then, it computes $E_K(M_b)$ and encrypts K under $\mathcal{T}$, and constructs the corresponding CT^*. If there exists $\mathcal{D}_k \in \mathcal{L}_D$ that can decrypt CT^*, then the challenge will be aborted. Finally, $\mathcal{C}$ outputs CT^*.*
  – ***Challenge 2:*** *$\mathcal{A}$ submits $(M_0, M_1, \mathcal{T}^*, i, j, m^*)$ with $m^* \neq m_{i,j}$ to $\mathcal{C}$ where M_0 and M_1 are two equal-length messages, $\mathcal{T}^*$ is an access tree structure, and (i,j,m^*) is used to ask $\mathcal{C}$ to update the value of user i's attribute j to be m^*. $\mathcal{C}$ will abort this challenge if the attribute j with value m^* is not a necessary attribute and value[1] in $\mathcal{T}^*$. $\mathcal{C}$ performs **Updating**(i,j,m^*) without returning the corresponding updated private key $\mathcal{D}_i^*$ to $\mathcal{A}$, but PK will be updated. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$ and $K \in G_T$ and constructs CT^* by computing $E_K(M_b)$ and using the updated PK to encrypt K under $\mathcal{T}^*$. If there exists $\mathcal{D}_k \in \mathcal{L}_D - \{\mathcal{D}_i^*\}$ that can decrypt CT^*, the challenge will be aborted. Finally, C outputs CT^*.*
- ***Phase 2: Phase 1*** *is repeated by $\mathcal{A}$ with the restriction that $\mathcal{A}$ cannot query **Decryption**$'(CT^*)$.*
- ***Guess:*** *$\mathcal{A}$ outputs $b' \in \{0,1\}$ as a guess of b and wins the game if $b' = b$.*

We define the advantage of $\mathcal{A}$ winning the CCA_{DM} game as

$$Adv_{\Pi}^{CCA_{DM}}(\mathcal{A}) = |Pr[b' = b] - 1/2|.$$

In Definition 5, Challenge 1 implies the CCA security of the proposed scheme and Challenge 2 implies the CCA security of the Updating algorithm which allows the attacker to update the attribute value as what she/he wants.

Definition 6. *A ciphertext-policy ABE scheme Π is said to be CCA_{DM} secure if for any adversary $\mathcal{A}$, within polynomial time, $Adv_{\Pi}^{CCA_{DM}}(\mathcal{A})$ is negligible.*

3 Our Construction

First, we define some notations in Table 1.

3.1 The Proposed Scheme

Our proposed scheme consists of six algorithms which are **Setup**, **Enrollment**, **Leaving**, **Updating**, **Encryption**, and **Decryption**. First, KGC will initialize the environment of our attribute-based encryption system by performing the Setup algorithm. Then, a user can enroll in the system by running the Enrollment algorithm with KGC. Furthermore, the user can leave the system or update her/his attribute values by performing the Leaving algorithm or Updating algorithm, respectively. KGC will maintain a public board which contains the current public key PK.

1. An attribute j with value m is called a necessary attribute and value in an access tree structure $\mathcal{T}$ if $\mathcal{T}$ can be satisfied only when the attribute j is with value m.

TABLE 1
The Notations

Notation	Meaning
k	a security parameter
$\mathcal{U}$	the set of registered users
A	the set of all provided attributes
A_i	the set of attributes user i possesses
$att(N)$	the attribute associated with node N
$val(N)$	the attribute value associated with node N
$attr_{i,j}$	user i's attribute j
k_N	the threshold value associated with node N
d_N	the degree of the polynomial of node N
$parent(N)$	the parent node of node N
$index(N)$	the index associated with node N
E_K	a symmetric encryption function with key K
D_K	a symmetric decryption function with key K

Our protocol has an assumption that KGC must be a trusted party. In a single-authority attributed-based encryption scheme, KGC owns the master key and can decrypt each ciphertext. Some works [9], [13], [15] with multiple authorities might be able to solve this problem but they still need the assumption that there is no collusion among these authorities, and besides, they will be also accompanied by higher costs.

3.1.1 Setup

This algorithm contains the following steps.

- **Step 1:** On the input k, generate two additive groups G_0 and G_1, a multiplicative group G_T, a bilinear mapping $e: G_0 \times G_1 \rightarrow G_T$, and two generators P and Q of G_0 and G_1, respectively, where G_0, G_1, and G_T are with prime order q.
- **Step 2:** Randomly choose two elements $\alpha, \beta \in \mathbb{Z}_q^*$ and a hash function H: $\{0,1\}^* \rightarrow \mathbb{Z}_q^*$.
- **Step 3:** Generate two default users as follows:

1. Set $\mathcal{U} = \{0,1\}$ and randomly choose $2(|A|+1)$ elements $\{v_{i,j}\}_{\forall i \in \mathcal{U}, j \in A}$ and $\{t_i\}_{\forall i \in \mathcal{U}}$ in $\mathbb{Z}_q^*$.
2. Compute

$$\begin{cases} \{V_j = (\prod_{\forall i \in \mathcal{U}} v_{i,j})Q\}_{\forall j \in A} \\ \{\overline{v_{i,j}} = t_i \prod_{\forall k \neq i, k \in \mathcal{U}} v_{k,j}^{-1} + v_{i,j} \bmod q\}_{\forall i \in \mathcal{U}, \forall j \in A}. \end{cases}$$

- **Step 4:** The public parameter is $PK = (G_0, G_1, G_T, e, H, P, Q, U = e(P,Q)^{\alpha(\beta-1)}, e(P,Q)^{\alpha\beta}, \{V_j, \{\overline{v_{i,j}}\}_{\forall i \in \mathcal{U}}\}_{\forall j \in A}, \mathbb{V})$ where $\mathbb{V}$ is the version number of the public parameter PK. The master key of KGC is $MK = \alpha Q$.

The asymmetric bilinear mapping setting of e can be implemented by BN-Curve [21] which is with faster software implementations.

3.1.2 Enrollment

The input of the algorithm is $(A_i, (m_{i,j})_{\forall j \in A_i}, MK)$ where A_i is the attribute set of user i and $m_{i,j}$ is the attribute value of user i for attribute j in A_i. KGC performs the Enrollment algorithm as follows:

- **Step 1:** Randomly pick $(|A| + |A_i| + 2)$ elements, $\{r_i, t_i, \{v_{i,j}\}_{\forall j \in A}, \{r_{i,j}\}_{\forall j \in A_i}\}$, in $\mathbb{Z}_q^*$.

- **Step 2: Set** $\mathcal{U} = \mathcal{U} \cup \{i\}$.
- **Step 3:** Compute

$$\begin{cases} \{h_{i,j} = H(m_{i,j})\}_{\forall j \in A_i} \\ \{V_j = v_{i,j} V_j\}_{\forall j \in A} \\ \{\overline{v_{i,j}} = t_i \prod_{\forall k \neq i, k \in \mathcal{U}} v_{k,j}^{-1} + v_{i,j} \bmod q\}_{\forall j \in A} \\ \{\overline{v_{k,j}} = (\overline{v_{k,j}} - v_{k,j}) v_{i,j}^{-1} + v_{k,j} \bmod q\}_{\forall k \neq i, k \in \mathcal{U}, \forall j \in A}. \end{cases}$$

- **Step 4:** Give the following $(3|A_i| + 1)$ pieces of information to user i as her/his private key $\mathcal{D}_i$:

$$\begin{cases} D_i = \alpha Q + t_i r_i Q \\ \{D_{i,j} = v_{i,j}^{-1}(r_i P + r_{i,j} h_{i,j} P)\}_{\forall j \in A_i} \\ \{D'_{i,j} = t_i r_{i,j} P\}_{\forall j \in A_i} \\ \{D''_{i,j} = r_i P + r_{i,j} h_{i,j} P\}_{\forall j \in A_i}. \end{cases}$$

- **Step 5:** Increase $\mathbb{V}$ and update $(\{V_j, \{\overline{v_{i,j}}\}_{\forall i \in \mathcal{U}}\}_{\forall j \in A}, \mathbb{V})$ in PK.

3.1.3 Leaving

This algorithm will be invoked in the need of revoking the private key of some registered user u. KGC will increase $\mathbb{V}$ and update $(\{V_j, \{\overline{v_{i,j}}\}_{\forall i \in \mathcal{U}}\}_{\forall j \in A}, \mathbb{V})$ in PK as follows:

$$\begin{cases} \{V_j = v_{u,j}^{-1} V_j\}_{\forall j \in A} \\ \{\overline{v_{k,j}} = (\overline{v_{k,j}} - v_{k,j}) v_{u,j} + v_{k,j} \bmod q\}_{\forall k \neq u, k \in \mathcal{U}, \forall j \in A}. \end{cases}$$

Finally, set $\mathcal{U} = \mathcal{U} \backslash \{u\}$ and delete $\{\overline{v_{u,j}}\}_{\forall j \in A}$ in PK.

3.1.4 Updating

This algorithm will be invoked when a registered user i applies for updating her/his attribute j to be with a value $m'_{i,j}$ where attribute j can be an existent attribute the user has already had or a new one. KGC will perform the following updating process:

- **Step 1:** Randomly pick three elements $v'_{i,j}$, r'_i, and $r'_{i,j}$ in $\mathbb{Z}_q^*$.
- **Step 2:** Compute $h'_{i,j} = H(m'_{i,j})$.
- **Step 3:** Give

$$\begin{cases} D_i = \alpha Q + t_i r'_i Q \\ D_{i,j} = {v'_{i,j}}^{-1}(r'_i P + r'_{i,j} h'_{i,j} P) \\ \{D_{i,k} = v_{i,k}^{-1}(r'_i P + r_{i,k} h_{i,k} P)\}_{\forall k \in A_i \backslash \{j\}} \\ D'_{i,j} = t_i r'_{i,j} P \\ D''_{i,j} = r'_i P + r'_{i,j} h'_{i,j} P \\ \{D''_{i,k} = r'_i P + r_{i,k} h_{i,k} P)\}_{\forall k \in A_i \backslash \{j\}} \end{cases}$$

to user i.
- **Step 4:** Increase $\mathbb{V}$ and update $(V_j, \{\overline{v_{i,j}}\}_{\forall i \in \mathcal{U}}, \mathbb{V})$ in PK:

$$\begin{cases} V_j = v_{i,j}^{-1} v'_{i,j} V_j \\ \overline{v_{i,j}} = (\overline{v_{i,j}} - v_{i,j}) + v'_{i,j} \bmod q \\ \overline{v_{k,j}} = (\overline{v_{k,j}} - v_{k,j}) v_{i,j} {v'_{i,j}}^{-1} + v_{k,j} \bmod q, \forall k \in \mathcal{U} \backslash \{i\}. \end{cases}$$

3.1.5 Encryption

Any one can perform this algorithm to encrypt a message M. The procedures are shown as follows:

1) **Access tree structure construction.** For a given tree T_S, choose a polynomial for each node of T_S in a top-down manner which is described below:
- **For the root node** R: Randomly choose an element $s \in \mathbb{Z}_q^*$ and a polynomial q_R of degree $d_R = k_R - 1$ with $q_R(0) = s$, where k_R is the threshold value of the root node R. Then, assign a unique index number x for each child of the root node.
- **For each internal node** N **other than** R: Randomly choose a polynomial q_N of degree $d_N = k_N - 1$ with $q_N(0) = q_{parent(N)}(index(N))$, where k_N is the threshold value of node N. Then, assign a unique index number x for each child of node N.
- **For each leaf node** N_L: Randomly choose a polynomial q_{N_L} of degree 0 with $q_{N_L}(0) = q_{parent(N_L)}(index(N_L))$.

2) **Ciphertext generation:** The sender obtains the newest public parameters from KGC. Then she/he randomly selects $K \in G_T$ and lets $\mathcal{N}_L$ be the set of all leaf nodes in $\mathcal{T}$ where $\mathcal{T}$ is defined in Definition 2. The ciphertext is

$$\begin{aligned} CT = (\mathcal{T}, \tilde{C} &= e(P, Q)^{\alpha\beta s} K, C = sP, C' = U^s, \\ \overline{M} &= E_K(M), C_r = H(K || \overline{M})P, \\ \{C_N &= q_N(0) V_{att(N)}, \\ C'_N &= q_N(0) H(val(N))Q, \\ &\{\overline{v_{i,att(N)}}\}_{\forall i \in \mathcal{U}}\}_{\forall N \in \mathcal{N}_L}, \mathbb{V}) \end{aligned}$$

where $\{\{\overline{v_{i,att(N)}}\}_{\forall i \in \mathcal{U}}\}_{\forall N \in \mathcal{N}_L}$ are the system's public parameters that can be excluded from the ciphertext to reduce the size of the ciphertext. However, storing the public parameters in a central server of the system will consume more bandwidth because each receiver who wants to decrypt the ciphertext needs to download the public parameters with version number $\mathbb{V}$ from the central server. There will be a trade-off between the above two ways. For those applications which require few member status renewals, a receiver can download the public parameters with the version number from the center and keep a copy in her/his local storage. Once the version number she/he kept does not equal to that in the ciphertext she/he wants to decrypt, she/he connects to the center and updates her/his stored public parameters. Our scheme allows an encryptor to include (for the system with frequent member updating) or exclude (for the system with infrequent member updating) the public parameters in a ciphertext, which will be flexible for different application systems.

For example, if Bernie in Section 1 wants to encrypt a message for those doctors who specialize in melancholia or those patients suffered from melancholia, he can set the access rule as ("Career: Doctor" AND "Speciality: Melancholia") OR ("Mental Disorder: Melancholia"). The corresponding access tree structure $\mathcal{T}$ is illustrated in Fig. 1 and the ciphertext is

$$\begin{aligned} CT = (\mathcal{T}, \tilde{C} &= e(P, Q)^{\alpha\beta s} K, C = sP, C' = U^s, \\ \overline{M} &= E_K(M), C_r = H(K || \overline{M})P, \\ C_A &= q_A(0) V_{Career}, \\ C'_A &= q_A(0) H(\text{“Doctor”})Q, \\ &\{\overline{v_{i,Career}}\}_{\forall i \in \mathcal{U}}, C_B = q_B(0) V_{Speciality}, \\ C'_B &= q_B(0) H(\text{“Melancholia”})Q, \\ &\{\overline{v_{i,Speciality}}\}_{\forall i \in \mathcal{U}}, C_D = q_D(0) V_{MentalDisorder}, \\ C'_D &= q_D(0) H(\text{“Melancholia”})Q, \\ &\{\overline{v_{i,MentalDisorder}}\}_{\forall i \in \mathcal{U}}, \mathbb{V}) \end{aligned}$$

where $s \in \mathbb{Z}_q^*$ is a randomly chosen element and K is the session key selected from G_T at random. Bernie sets the polynomial $q_R(X)$ with degree 0 as $q_R(X) = s$ for node R and the polynomial $q_C(X) = s + c_r X$ with degree 1 where $q_R(v) = q_C(0) = s$ and c_r is randomly selected from $\mathbb{Z}_q^*$ for node C. Furthermore, $q_A(0) = q_C(x) = s + c_r x$, $q_B(0) = q_C(y) = s + c_r y$, and $q_D(0) = q_R(w) = s$ are set for leaf nodes A, B, and D respectively.

3.1.6 Decryption

We will specify our decryption procedure as a recursive algorithm. We first define a recursive algorithm $DecryptNode(CT, \mathcal{D}_i, N)$ that takes the ciphertext CT, the private key $\mathcal{D}_i$ of user i, and a node N in $\mathcal{T}$ as input where $\mathcal{D}_i$ is associated with a attribute set A_i and attribute values $(m_{i,j})_{\forall j \in A_i}$. $DecryptNode(CT, \mathcal{D}_i, N)$ is defined below, where we let $V_j = v_j P$.

- **If N is a leaf node:** Let $j = att(N)$. If j is in A_i and $m_{i,j} = val(N)$, then

$$\begin{aligned}
&DecryptNode(CT, \mathcal{D}_i, N)\\
&= \frac{e(D_{i,j}, \overline{v_{i,j}} C_N)}{e(D'_{i,j}, C'_N) e(D''_{i,j}, C_N)}\\
&= \frac{e(v_{i,j}^{-1}(r_i P + r_{i,j} h_{i,j} P), (t_i v_j^{-1} v_{i,j} + v_{i,j}) q_N(0) v_j Q)}{e(t_i r_{i,j} P, q_N(0) h_{i,j} Q) e(r_i P + r_{i,j} h_{i,j} P, v_j q_N(0) Q)}\\
&= \frac{e(v_{i,j}^{-1}(r_i P + r_{i,j} h_{i,j} P), t_i v_{i,j} q_N(0) Q + v_{i,j} v_j q_N(0) Q)}{e(t_i r_{i,j} P, q_N(0) h_{i,j} Q) e(r_i P + r_{i,j} h_{i,j} P, v_j q_N(0) Q)}\\
&= \frac{e(r_i P + r_{i,j} h_{i,j} P, t_i q_N(0) Q) e(r_i P + r_{i,j} h_{i,j} P, v_j q_N(0) Q)}{e(t_i r_{i,j} P, q_N(0) h_{i,j} Q) e(r_i P + r_{i,j} h_{i,j} P, v_j q_N(0) Q)}\\
&= \frac{e(r_i P, t_i q_N(0) Q) e(r_{i,j} h_{i,j} P, t_i q_N(0) Q)}{e(t_i r_{i,j} P, q_N(0) h_{i,j} Q)}\\
&= e(P, Q)^{t_i r_i q_N(0)}.
\end{aligned}$$

  Otherwise, $DecryptNode(CT, \mathcal{D}_i, N) = \perp$.

- **If N is an internal node**: For each child node N_c of N, call $DecryptNode(CT, \mathcal{D}_i, N_c)$ and store the output as F_{N_c}. Let $\mathcal{I}_c$ be a k_N-sized set which contains the indexes of the child nodes N_c's such that $F_{N_c} \neq \perp$ for each N_c. If no such set exists, the node N is not satisfied and thus $DecryptNode(CT, \mathcal{D}_i, N) = \perp$. Otherwise, return

$$\begin{aligned}
F_N &= \prod\nolimits_{\forall index(N_c) = z \in \mathcal{I}_c} F_{N_c}^{\Delta_{z,\mathcal{I}_c}(0)}\\
&= \prod\nolimits_{\forall index(N_c) = z \in \mathcal{I}_c} \left(\varpi^{t_i r_i q_{N_c}(0)}\right)^{\Delta_{z,\mathcal{I}_c}(0)}\\
&= \prod\nolimits_{\forall index(N_c) = z \in \mathcal{I}_c} \left(\varpi^{t_i r_i q_{parent(N_c)}(z)}\right)^{\Delta_{z,\mathcal{I}_c}(0)}\\
&= \prod\nolimits_{\forall index(N_c) = z \in \mathcal{I}_c} \left(\varpi^{t_i r_i q_N(z)}\right)^{\Delta_{z,\mathcal{I}_c}(0)}\\
&= \varpi^{t_i r_i q_N(0)}
\end{aligned}$$

  where $\varpi = e(P, Q)$.

After computing $DecryptNode(CT, \mathcal{D}_i, R)$, the algorithm returns

$$\begin{aligned}
A &= DecryptNode(CT, \mathcal{D}_i, R)\\
&= e(P, Q)^{t_i r_i q_R(0)}\\
&= e(P, Q)^{t_i r_i s}
\end{aligned}$$

where R is the root node. Finally, the session key K can be obtained by computing $A \cdot \tilde{C} / (e(C, D_i) \cdot C')$ where

$$\begin{aligned}
\frac{A \cdot \tilde{C}}{e(C, D_i) \cdot C'} &= \frac{e(P, Q)^{t_i r_i s} \cdot e(P, Q)^{\alpha\beta s} K}{e(sP, (\alpha + t_i r_i) Q) \cdot e(P, Q)^{\alpha(\beta - 1)s}}\\
&= \frac{e(P, Q)^{\alpha\beta s} \cdot e(P, Q)^{t_i r_i s} K}{e(P, Q)^{(\alpha s + \alpha(\beta - 1)s + t_i r_i s)}}\\
&= \frac{e(P, Q)^{\alpha\beta s + t_i r_i s} K}{e(P, Q)^{\alpha\beta s + t_i r_i s}}\\
&= K.
\end{aligned}$$

The decryption procedure will return $\perp$ if $C_r \neq H(K || \overline{M}) P$; Otherwise, it returns M by computing $M = D_K(\overline{M})$.

Following the example shown in the end of Section 3.1.5, we assume that there is a user, i, whose attributes are ("Career: Doctor" and "Speciality: Melancholia"). Her/His secret keys are (D_i, $D_{i,Carrer}$, $D_{i,Speciality}$, $D'_{i,Carrer}$, $D'_{i,Speciality}$, $D''_{i,Carrer}$, $D''_{i,Speciality}$). She/He satisfies the access tree $\mathcal{T}$ and she/he can decrypt the ciphertext CT by the following procedures. First, for the leaf nodes A and B in Fig. 1 she/he computes

$$\begin{cases}
\mathbb{A} = \frac{e(D_{i,Carrer}, \overline{v_{i,Carrer}} C_A)}{e(D'_{i,Carrer}, C'_A) e(D''_{i,Carrer}, C_A)}\\
\mathbb{B} = \frac{e(D_{i,Speciality}, \overline{v_{i,Speciality}} C_B)}{e(D'_{i,Speciality}, C'_B) e(D''_{i,Speciality}, C_B)}
\end{cases}$$

and obtains $\mathbb{A} = e(P, Q)^{t_i r_i q_A(0)}$ and $\mathbb{B} = e(P, Q)^{t_i r_i q_B(0)}$. Then, she/he computes $e(P, Q)^{t_i r_i s} = e(P, Q)^{t_i r_i q_R(0)} = e(P, Q)^{t_i r_i q_C(0)} = \mathbb{A}^{\frac{-y}{x-y}} \times \mathbb{B}^{\frac{-x}{y-x}}$ according to Lagrange interpolation where (x, y) are the indexes of nodes A and B, respectively. Finally, she/he obtains the session key K by computing

$$K = \frac{e(P, Q)^{t_i r_i s} \tilde{C}}{e(C, D_i) C'}.$$

3.2 Dynamic Membership

As mentioned in Definition 4, a general ABE scheme with dynamic membership should satisfy Revocability, Renewability, Expandability, and Independence. Our proposed scheme achieves Expandability via the *Enrollment* algorithm. Revocability and Renewability are implemented by the *Leaving* and *Updating* algorithms, respectively. Furthermore, when a user performs the *Leaving* or *Updating* algorithm, the other users do not need to update their private keys. It turns out that our scheme meets the Independence property.

3.3 Performance Evaluation on Encryption and Decryption

In this subsection, we introduce the performance evaluation on encryption and decryption which are performed by users. In our scheme, a user needs $(2m_c + 3)GM + 1EK + (m_c + 1)H$ operations to encrypt a message and $2(2m_c + 1)GM + (3m_c + 1)e + 1DK$ operations to decrypt a ciphertext where m_c is the number of the attributes associated to the ciphertext. GM, EK/DK, e, and H are defined in Table 2. From the benchmark of Crypto++ [22] and pairing based cryptography [23], [24], the execution time of each operation is provided in

TABLE 2
Notations for Performance Evaluation

Notation	Meaning	Time(millisecond)
GM	multiplication in G_0 or exponentiation in G_1	125
e	bilinear mapping	262
EK/DK	symmetric encryption/decryption, (AES-CBC-256)	8.8
H	hash function (SHA-256)	7

Table 2 where the benchmark is derived from [22]–[24] on a PC with Intel Core 2 2.4 GHz processor.

Therefore, the execution times of the example in Section 3.1.5 for encrypting the session key K is $(9GM + 4H) = 1.153$ seconds. And the sender needs 8.8 milliseconds per Mbit to encrypt the plaintext M. The decryption time of the example in Section 3.1.6 to retrieve the session key is $(14GM + 10e) = 4.37$ seconds. When a receiver retrieves a session key, she/he needs 8.8 milliseconds per Mbit to decrypt the ciphertext $\overline{M}$.

4 Security Proof

We have defined the CCA_{DM} game in Definition 5 under the chosen-ciphertext attack model. Now, we demonstrate the security of the proposed scheme by Theorem 4.1 and show the proof without random oracles in the following.

Theorem 4.1. *The proposed CP-ABE-DM scheme is CCA_{DM} secure under the DBDH assumption.*

Proof. C who tries to solve the DBDH problem will be a challenger to simulate the CCA_{DM} game. Given a DBDH instance $(G_0, G_1, G_T, q, P, Q, e, aP, bP, cP, aQ, bQ, cQ, Z)$, the challenger $\mathcal{C}$ follows Step 3 of the Setup algorithm in Section 3.1.1 to generate a user set $\mathcal{U} = \{0, 1\}$ and $\{V_j, \{\overline{v_{i,j}}\}_{\forall i\in\mathcal{U}}\}_{\forall j\in A}$. Then, $\mathcal{C}$ publishes $PK = (G_0, G_1, G_T, e, H, P, Q, U = e(aP, bQ - Q), e(aP, bQ), \{V_j, \{\overline{v_{i,j}}\}_{\forall i\in\mathcal{U}}\}_{\forall j\in A}, \mathbb{V})$ and sets the master key $MK = aQ$. $\mathcal{C}$ creates a list $\mathcal{L}_{\mathcal{D}}$, and simulates the oracles in Phase 1 of the CCA_{DM} game as follows.

- *Enrollment*(S_i): Due to owning aQ, $\mathcal{C}$ can follow the Enrollment algorithm in Section 3.1.2 to generate the private key $\mathcal{D}_i$ according to S_i and update PK. $\mathcal{C}$ will return $\mathcal{D}_i$ and store $\mathcal{D}_i$ in $\mathcal{L}_{\mathcal{D}}$.
- *Leaving*(i): $\mathcal{C}$ follows the Leaving algorithm in Section 3.1.3 and then removes $\mathcal{D}_i$ from $\mathcal{L}_{\mathcal{D}}$.
- *Updating*(i, j, m): Since $\mathcal{C}$ has aQ, it can follow the Updating algorithm in Section 3.1.4 and refreshes the updated parts of $\mathcal{D}_i$ in $\mathcal{L}_{\mathcal{D}}$.
- *Decryption*$'(CT)$: On receiving $CT = (\mathcal{T}, \tilde{C}, C, C', \overline{M}, C_r, \{C_N, C'_N, \{\overline{v_{i,att(N)}}\}_{\forall i\in\mathcal{U}}\}_{\forall N\in N_L}, \mathbb{V})$ where N_L is the set of all leaf nodes in $\mathcal{T}$. Let $A_{\mathcal{T}} = \{att(N) | N \in N_L\}$ and $m_{att(N)} = val(N)$ for each $N \in N_L$. $\mathcal{C}$ first checks if there exists a private key $\mathcal{D}_i$ in $\mathcal{L}_{\mathcal{D}}$ which can decrypt CT. If so, $\mathcal{C}$ decrypts CT by using $\mathcal{D}_i$ to perform the Decryption algorithm in Section 3.1.6 and then returns the output to $\mathcal{A}$. Otherwise, $\mathcal{C}$ runs the following procedures to create a pseudo user w, and then $\mathcal{C}$ takes user w's private key to decrypt CT.

```
$Cipher(\mathcal{T}, N, \mathbb{C}_0)$
1. Locate the node $N$ in $\mathcal{T}$;
2. If ($N$ is a leaf node ) {
3.     Set $j = att(N)$ and $m = val(N)$;
4.     Compute $u_m = H(m)$;
5.     Compute $C_N = v_j\mathbb{C}_0$; // $v_j = \prod_{\forall i\in\mathcal{U}} v_{i,j}$
6.     Compute $C'_N = u_m\mathbb{C}_0$;
7.     Store $(C_N, C'_N)$ in $\mathcal{L}_C$; }
8. Else {
9.     Randomly select $k_N - 1$ elements $c_i \in \mathbb{Z}_q^*$;
10.    For each child $N_C$ of the node $N$ {
11.        Set $x = index(N_C)$;
12.        If ($k_N - 1 > 0$ ) {
13.            Compute $\overline{\mathbb{C}_0} = \mathbb{C}_0 + \sum_{i=1}^{k_N-1} c_i x^i Q$; }
14.        Else {
15.            Set $\overline{\mathbb{C}_0} = \mathbb{C}_0$; }
16.        Call $Cipher(\mathcal{T}, N_C, \overline{\mathbb{C}_0})$; } }
```

Fig. 2. The $Cipher$ algorithm.

1) Randomly pick $(2|A_{\mathcal{T}}| + 2)$ elements which are

$$\begin{cases} r_w, t_w \in \mathbb{Z}_q^* \\ \{v_{w,j}, r_{w,j}\}_{\forall j\in A_{\mathcal{T}}} \in \mathbb{Z}_q^*. \end{cases}$$

2) Compute

$$\begin{cases} \{h_{w,j} = H(m_j)\}_{\forall j\in A_{\mathcal{T}}} \\ \{C_N = v_{w,att(N)} C_N\}_{\forall N\in N_L} \\ \{\overline{v_{w,j}} = t_w \prod_{\forall k\in\mathcal{U}} v_{k,j}^{-1} + v_{w,j} \bmod q\}_{\forall j\in A_{\mathcal{T}}} \end{cases}$$

where C_N is retrieved from CT and refreshed in this step corresponding to the enrollment of the pseudo user w.

3) Create the pseudo user w's private key $\mathcal{D}_w = (D_w, \{D_{w,j}\}, \{D'_{w,j}\}, \{D''_{w,j}\})$ which is computed as follows.

$$\begin{cases} D_w = aQ + t_w r_w Q \\ \{D_{w,j} = v_{w,j}^{-1}(r_w P + r_{w,j} h_{w,j} P)\}_{\forall j\in A_{\mathcal{T}}} \\ \{D'_{w,j} = t_w r_{w,j} P\}_{\forall j\in A_{\mathcal{T}}} \\ \{D''_{w,j} = r_w P + r_{w,j} h_{w,j} P\}_{\forall j\in A_{\mathcal{T}}}. \end{cases}$$

4) Use $\mathcal{D}_w$ to decrypt CT by performing the Decryption algorithm in Section 3.1.6, and then return the output to $\mathcal{A}$.

- *Challenge:*
  - *Challenge* 1: After receiving $(M_0, M_1, \mathcal{T}^*)$, $\mathcal{C}$ sets $\mathcal{N}_L$ to be the set of all leaf nodes of $\mathcal{T}^*$ and creates list $\mathcal{L}_C$. Let R be the root node of $\mathcal{T}^*$. Then $\mathcal{C}$ calls $Cipher(\mathcal{T}^*, R, cQ)$ which is a recursive algorithm defined in Fig. 2 and is used to generate and store $\{C_N, C'_N\}_{\forall N\in\mathcal{N}_L}$ in $\mathcal{L}_C$. After performing the $Cipher$ algorithm, $\mathcal{C}$ randomly selects $b'' \in \{0, 1\}$ and $K \in G_T$ and prepares $CT^* = (\mathcal{T}^*, \tilde{C} = Z \cdot K, C = cP, C' = \frac{Z}{e(aP, cQ)}, \overline{M} = E_K(M_{b''}), C_r = H(K||\overline{M})P, \{C_N, C'_N, \{\overline{v_{i,att(N)}}\}_{\forall i\in\mathcal{U}}\}_{\forall N\in\mathcal{N}_L}, \mathbb{V})$ where $\{C_N, C'_N\}_{\forall N\in\mathcal{N}_L}$ are retrieved from $\mathcal{L}_C$. If there exists $\mathcal{D}_k \in \mathcal{L}_D$ that can decrypt CT^*, the challenge will be aborted. Finally, $\mathcal{C}$ returns CT^* to $\mathcal{A}$.

TABLE 3
Feature Comparisons

Scheme	Dynamic Membership		ASA	Special Feature
	Updating	Leaving		
Ours	Yes	Yes	Yes	No Private Key Refreshment
[4]	No	No	Yes	Direct and Indirect Revocation
[5]	No	Yes	No	Multi-Authority
[12]	No	No	No	Dual Policy
[13]	No	No	No	Multi-Authority
[15]	No	No	No	Multi-Authority
[7]	No	No	No	Full Logic Expression
[8]	No	No	No	Key Delegation
[9]	No	No	No	Multi-Authority
[10]	No	No	No	-
[11]	No	No	No	-
[14]	No	No	No	-
[25]	No	No	No	Full Logic Expression
[26]	No	No	No	
[27]	No	No	No	Multi-Authority
[28]	No	No	No	Multi-Authority
[29]	No	No	No	Attribute Hierarchy
[30]	No	No	Yes	Unbounded Attribute
[31]	No	No	No	Constant Ciphertext Size
[32]	No	No	No	Multi-Authority
[33]	No	Yes	No	-
ASA: Arbitrary-State Attribute				

TABLE 4
Notations for Performance Comparisons

Notation	Meaning
n	the number of the members in an ABE system
m	the number of the attributes provided in an ABE system
m_c	the number of attributes associated to a ciphertext
m_d	the maximum of $\|Cover(R)\|$ in [4] where R is the set of revoked users and $m_d < m$
m_u	the number of a user's attributes
m_a	the number of authorities
m_{or}	the number of OR gates of the access rule associated to a ciphertext
m_{and}	the number of the AND conjunction attributes of the access rule associated to a ciphertext
m_{not}	the number of the NOT conjunction attributes of the access rule associated to a ciphertext
DR, IR	Direct Mode and Indirect Mode in [4], respectively
SA, OA	Subject Attribute and Objected Attribute in [12], respectively

– *Challenge* 2: After receiving $(M_0, M_1, \mathcal{T}^*, i, j, m^*)$ with $m^* \neq m_{i,j}$, $\mathcal{C}$ checks if the attribute j with value m^* is a necessary attribute and value in $\mathcal{T}^*$. If not, $\mathcal{C}$ will abort the challenge. Otherwise, $\mathcal{C}$ sets $\mathcal{N}_L$ to be the set of all leaf nodes of $\mathcal{T}^*$ and creates a list $\mathcal{L}_C$. Then, $\mathcal{C}$ performs *Updating*(i, j, m^*), but the updated private key $\mathcal{D}_i^*$ is not returned to $\mathcal{A}$. After PK has been updated, $\mathcal{C}$ calls $Cipher(\mathcal{T}^*, R, cQ)$. $\mathcal{C}$ retrieves $\{C_N, C'_N\}_{\forall N \in \mathcal{N}_L}$ from $\mathcal{L}_C$, randomly selects $b'' \in \{0,1\}$ and $K \in G_T$, and computes $CT^* = (\mathcal{T}^*,\ \tilde{C} = Z \cdot K,\ C = cP,\ C' = \frac{Z}{e(aP,cQ)},\ \overline{M} = E_K(M_{b''}),\ C_r = H(K||\overline{M})P,\ \{C_N, C'_N,\ \{\overline{v_{i,att(N)}}\}_{\forall i \in \mathcal{U}}\}_{\forall N \in \mathcal{N}_L},\ \mathbb{V})$. If there exists $\mathcal{D}_k \in \mathcal{L}_D - \{\mathcal{D}_i^*\}$ that can decrypt CT^*, the challenge will be aborted. Finally, $\mathcal{C}$ returns CT^* to $\mathcal{A}$.

Phase 2: In Phase 2, all oracles can be simulated as those in Phase 1. However, if $\mathcal{A}$ queries $Decryption'(CT^*)$, $\mathcal{C}$ will abort it.

- *Guess*: $\mathcal{A}$ outputs $b''' \in \{0,1\}$. If $b''' = b''$, $\mathcal{C}$ outputs $b' = 1$. Otherwise, $\mathcal{C}$ outputs $b' = 0$. If adversary $\mathcal{A}$, within polynomial time, can win the CCA_{DM} game with non-negligible advantage at least ϵ, $\mathcal{C}$ has non-negligible advantage ϵ' to solve the DBDH problem where $\epsilon' \geq \epsilon - \sigma$ and σ is a negligible advantage. The probability analysis is shown below.
- If $Z = e(P,Q)^{abc}$, we get $Pr[\mathcal{C}(P, Q, aP, bP, cP, aQ, bQ, cQ, e(P,Q)^{abc}) = 1] = Pr[b''' = b'']$ where $|Pr[b''' = b''] - \frac{1}{2}| \geq \epsilon$. Otherwise (when Z is randomly chosen from G_T), $Pr[\mathcal{C}(P, Q, aP, bP, cP, aQ, bQ, cQ, Z) = 1] = Pr[b''' = b'']$ with $|Pr[b''' = b''] - \frac{1}{2}| \leq \sigma$ where σ is the advantage of breaking the semantic security of E_K. Therefore, we have

$$\begin{aligned}
&|Pr[\mathcal{C}(P, Q, aP, bP, cP, aQ, bQ, cQ, e(P,Q)^{abc}) = 1] \\
&- Pr[\mathcal{C}(P, Q, aP, bP, cP, aQ, bQ, cQ, Z) = 1]| \\
&\geq |(\frac{1}{2} \pm \epsilon) - (\frac{1}{2} \pm \sigma)| \\
&\geq \epsilon - \sigma.
\end{aligned}$$

□

5 Comparisons

In this section, we discuss some crucial features which will affect the flexibility and practicability of an ABE system. Besides, we analyze the performance on computation, storage, and communication.

- **Dynamic Membership**: This feature includes Enrollment, Updating, and Leaving, which have been defined in Definition 3. It allows an ABE system to manage member enrollment, attribute and value updating, and member revocation efficiently.
- **Sender Updating**: Due to possible membership variation, a sender has to grab the newest public information which is maintained in a public place before she/he encrypts a message.
- **Receiver Updating**: Due to possible membership variation, a receiver has to interact with the system to refresh her/his private key or retrieve the newest public information before she/he tries to decrypt a ciphertext. Receiver updating is much less efficient than sender updating for users when dealing with membership variation.
- **No Private Key Refreshment**: Members do not have to interact with the system to refresh their private keys when the membership or any of the members' attributes has been changed.
- **Arbitrary-State Attribute**: The domain of each attribute is a variable-length string, not a binary bit only.

Consider the features of **Sender Updating** and **Receiver Updating**. For an ABE system with attribute updating or member revocation, it should be with at least one of these two features to inject the newest information of member

TABLE 5
Performance Comparisons: Computation Cost

	Encryption Cost of the Sender	Decryption Cost of the Receiver	The Necessary Computation Cost of the Center		
			Enrollment	Updating	Leaving
Ours	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m + m_u)$	$\mathcal{O}(m + m_u)$	$\mathcal{O}(m)$
[4]	$\mathcal{O}(m \times m_c + m_d)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m \times m_p \times m_u)$	$\mathcal{O}(n \times m_p \times m_u \times m)$	DR: $\mathcal{O}(1)$, IR: $\mathcal{O}(n \times m_d^2)$
[5]	$\mathcal{O}(m)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$	$\mathcal{O}(1)$
[12]	$\mathcal{O}(m \times m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m \times m_u)$	SA: $\mathcal{O}(n \times m)$, DA: $\mathcal{O}(n \times m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$
[13]	$\mathcal{O}(m_{and} \times m_{or})$	$\mathcal{O}(m_c)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n)$	$\mathcal{O}(n \times m_u)$
[15]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m^2)$	$\mathcal{O}(n \times m^2)$	$\mathcal{O}(n \times m^2)$
[7]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c + m_u)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[8]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(n \times m)$	$\mathcal{O}(n \times m)$
[9]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c \times m_L)$	$\mathcal{O}(m)$	$\mathcal{O}(n)$	$\mathcal{O}(n \times m)$
[10]	$\mathcal{O}(m_{and} + m_{not})$	$\mathcal{O}(m_u + m_{not})$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[11]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n)$	$\mathcal{O}(n \times m_u)$
[14]	$\mathcal{O}(m \times m_c + m_c^2)$	$\mathcal{O}(m \times m_c m_c^2)$	$\mathcal{O}(m_u^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n \times m_u^2)$
[25]	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[26]	$\mathcal{O}(m)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$	$\mathcal{O}(n \times m)$	$\mathcal{O}(n \times m)$
[27]	$\mathcal{O}(m)$	$\mathcal{O}(m_c \times m_u)$	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$
[28]	$\mathcal{O}(m_a + m_c)$	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$
[29]	$\mathcal{O}(m_c^2)$	$\mathcal{O}(m_c^2)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[30]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[31]	$\mathcal{O}(m)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$
[32]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c^2)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[33]	$\mathcal{O}(m_c)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$

statuses into encryption and/or decryption. When the statuses of members are changed, either the public information or the private keys of users must be updated accordingly. As mentioned in [4], the communication between all receivers and the ABE system center may become a bottleneck because all of the users, who are possible receivers, have to connect to the center to refresh their private keys, where the center needs to take additional computation to deal with the requests, or retrieve newest public information in those schemes which need Receiver Updating. In our scheme, if the sender does not include the public information in a ciphertext, only the receivers who have no updated public information should retrieve it from the center. More importantly, the receivers in our scheme are free from updating their private keys that satisfies the feature, No Private Key Refreshment.

In Table 3, only our proposed scheme satisfies complete Dynamic Membership and supports both of the features of No Private Key Refreshment and Arbitrary-State Attribute. The scheme [30] possesses the feature of Unbounded Attribute so it can satisfy the feature of Arbitrary-State Attribute as ours. The schemes [4] and [5] provide mechanisms to handle member leaving.

In Tables 5 and 6, we take big O notation to express the computation and communication costs where the used symbols are defined in Table 4. In Table 5, we evaluate the computation cost of sender's encryption and receiver's decryption. In Table 5, we provide comparisons on the size of user's private key, ciphertext, and the system's public information. Furthermore, we also investigate the necessary computations and communications performed by the center in each of the other schemes when dealing with dynamic membership.

In order to simplify the comparisons, we assume that there is only one authority in [5], [13], [15], [9], [27], [28], [32], and the access trees in [8], [11], [14], and our scheme are complete binary trees. Besides, a user must satisfy all of the attributes of the access tree structure associated to a ciphertext when she/he wants to decrypt it.

Consider dynamic membership management. The center generates a private key for an enrolling user. When a user leaves the system or updates her/his attributes or values, the center has to revoke the user's old private key by updating the public information. In those schemes which do not possess a mechanism tailored for dealing with users' leaving and attribute updating, all of the other users require connecting the center and refresh their private keys. This will be very inefficient in membership management. Whatever enrollment, leaving, or attribute updating our system performs, at most one user connects the center to update her/his private key. In Tables 5 and 6, we can observe that the computation and communication costs in all of the other schemes are linearly increased by n, i.e., the number of the members, when handling member leaving and attribute updating.

The schemes, except ours, includes n to handle member leaving and attribute updating, i.e., the complexity is related to the number of the members in an ABE system.

Table 6 shows the size of a ciphertext under the assumption that the membership is not changed. In our scheme, once the membership is changed, only the receivers who want to decrypt the ciphertext and have no up-to-date public parameters need to download the parameters required for updating the out-of-date part of the public parameters from the center instead of downloading all of the public parameters. The communication cost is much less than the cost of that all

TABLE 6
Performance Comparisons: Storage and Communication Cost

	Size of Private Key	Size of Ciphertext	Size of Public Parameters	The Communication Cost (between the center and a user)		
				Enrollment	Updating	Leaving
Ours	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(n \times m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(m_u)$	$\mathcal{O}(1)$
[4]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(n + m)$	$\mathcal{O}(m_u \times m_p)$	$\mathcal{O}(n \times m_u \times m_p)$	DR: $\mathcal{O}(1)$, IR: $\mathcal{O}(m_d)$
[5]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(n + m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(1)$
[12]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(n + m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[13]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(n + m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[15]	$\mathcal{O}(m)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m^2)$	$\mathcal{O}(n \times m^2)$	$\mathcal{O}(n \times m^2)$
[7]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[8]	$\mathcal{O}(m)$	$\mathcal{O}(m_c)$	$\mathcal{O}(1)$	$\mathcal{O}(m)$	$\mathcal{O}(n \times m)$	$\mathcal{O}(n \times m)$
[9]	$\mathcal{O}(m)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[10]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_{and} + m_{or})$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[11]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n)$	$\mathcal{O}(n \times m_u)$
[14]	$\mathcal{O}(m_L^2 \times m_u)$	$\mathcal{O}(m_L \times m_c)$	$\mathcal{O}(m \times m_L)$	$\mathcal{O}(m_L^2 \times m_u)$	$\mathcal{O}(n \times m_L)$	$\mathcal{O}(n \times m_L \times m_u)$
[25]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[26]	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(n \times m)$	$\mathcal{O}(n \times m)$
[27]	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(m_a \times m_c)$	$\mathcal{O}(m \times m_a)$	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$
[28]	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(m_a \times m_c)$	$\mathcal{O}(n \times m_a \times m)$	$\mathcal{O}(m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$	$\mathcal{O}(n \times m_a \times m_u)$
[29]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[30]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(1)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[31]	$\mathcal{O}(m \times m_u)$	$\mathcal{O}(1)$	$\mathcal{O}(m)$	$\mathcal{O}(m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$	$\mathcal{O}(n \times m \times m_u)$
[32]	$\mathcal{O}(m_u)$	$\mathcal{O}(m_c)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$
[33]	$\mathcal{O}(m_u)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$\mathcal{O}(m_u)$	$\mathcal{O}(n \times m_u)$	$\mathcal{O}(n \times m_u)$

users connect to the center to update their private keys. For the other schemes, except [4] and [5] that can deal with user leaving, all of the users, including the receivers who want to decrypt the ciphertext, have to update their private keys and public parameters whenever the membership of any user is changed.

The schemes of [5], [13], [15], [9], [27], [28], [32] support multiple authorities which share the computation and communication loading of the center. However, the communication cost of those schemes is not reduced and furthermore it is less convenient for users because they must communicate with many authorities to access updated public information and refresh their private keys. Besides, in Table 5, the center of the scheme [5] does not perform any computation to deal with user revocation. However, it requires to maintain a valid user set S_u to record the current valid users and S_u will be injected to the ciphertext. Nevertheless, the set S_u will break the user privacy, i.e., every one can learn the identities of the users who can decrypt the ciphertext.

6 Conclusion

In this paper, we proposed a ciphertext-policy attribute-based encryption scheme with dynamic membership. A user is allowed to enroll and leave from an ABE system, and she/he can also change her/his attributes and the values corresponding to the attributes. It is unnecessary for anyone else to update her/his private key when enrollment, leaving, or attribute updating occurs. In addition, to the best of our knowledge, our scheme is the first ABE scheme which can support arbitrary-state attributes and attribute (and value) updating with Sender Updating Only. These advantages will make an ABE service more efficient and flexible for practical applications. Finally, we have also formally proved the security of our scheme under the CCA model.

Acknowledgment

This work was partially supported by the National Science Council of the Taiwan under grants NSC 101-2219-E-110-003, NSC 101-2219-E-110-005, and "Aim for the Top University Plan" of the National Sun Yat-sen University and Ministry of Education, Taiwan, R.O.C.

References

[1] A. Fiat and M. Naor, "Broadcast encryption," in *Proc. Int. Cryptol. Conf. (CRYPTO'93)*, 1994, vol. 773, pp. 480–491.
[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Adv. Cryptol. Conf. (EUROCRYPT'05)*, 2005, vol. 3494, pp. 557–557.
[3] J. Baek, W. Susilo, and J. Zhou, "New constructions of fuzzy identity-based encryption," in *Proc. 2nd ACM Symp. Inf. Comput. Commun. Secur.*, 2007, pp. 368–370.
[4] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Proc. Int. Conf. Cryptography Coding*, 2009, vol. 5921, pp. 278–300.
[5] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proc. Int. Conf. Pairing-Based Cryptography Pairing*, 2009, vol. 5671, pp. 248–265.
[6] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, 2008 pp. 417–426.
[7] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.

[8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[9] M. Chase, "Multi-authority attribute based encryption," *Theory Cryptography*, vol. 4392, pp. 515–534, 2007.

[10] D. Lubicz, and T. Sirvent, "Attribute-based broadcast encryption scheme made efficient," in *Proc. Cryptol. Africa 1st Int. Conf. Progr. Cryptol.*, 2008, pp. 325–342.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[12] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," *Appl. Cryptography Netw. Secur.*, vol. 5536, pp. 168–185, 2009.

[13] S. Mller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Proc. Inf. Secur. Cryptol. (ICISC'08)*, 2009, vol. 5461, pp. 20–36.

[14] X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably secure and efficient bounded ciphertext policy attribute based encryption," in *Proc. 4th Int. Symp. Inf. Comput. Commun. Secur.*, 2009, pp. 343–352.

[15] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 121–130.

[16] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Adv. Cryptol. (CRYPTO'01)*, 2001, vol. 2139, pp. 213–229.

[17] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Proc. Adv. Cryptol. (CRYPTO'99)*, 1999, vol. 1666, p. 79.

[18] B. Archer and E. W. Weisstein, "Lagrange Interpolating Polynomial," [Online]. Available: http://mathworld.wolfram.com/Lagrange-InterpolatingPolynomial.html.

[19] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, pp. 612–613, 1979.

[20] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Adv. Cryptol. (CRYPTO'01)*, 2001, vol. 2139, pp. 213–229.

[21] A. Devegili, M. Scott, and R. Dahab, "Implementing cryptographic pairings over barreto-naehrig curves," in *Pairing-Based Cryptography Pairing 2007*, vol. 4575, T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, Eds. Berlin, Germany: Springer, 2007, pp. 197–207.

[22] Crypto++. (2009). "Crypto++ 5.6.0 benchmark," [Online]. Available: http://www.cryptopp.com/benchmarks-amd64.html.

[23] A. Ramachandran, Z. Zhou, and D. Huang, "Computing cryptographic algorithms in portable and embedded devices," in *Proc. IEEE Int. Conf. Portable Inf. Devices (PORTABLE'07)*, May 2007, pp. 1–7.

[24] X. Xiong, D. Wong, and X. Deng, "TinyPairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks," in *Proc. 2010 IEEE Wireless Commun. Netw. Conf. (WCNC)*, April 2010, pp. 1–6.

[25] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography (PKC'11)*, 2011, pp. 53–70.

[26] S. Yu, K. Ren, and W. Lou, "Attribute-based on-demand multicast group setup with membership anonymity," *Comput. Netw.*, vol. 54, pp. 377–386, 2010.

[27] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Inf. Sci.*, vol. 180, pp. 2618–2632, 2010.

[28] V. Boovi, D. Socek, R. Steinwandt, and V. I. Villnyi, "Multi-authority attribute-based encryption with honest-but-curious central authority," *Int. J. Comput. Math.*, vol. 89, no. 3, pp. 268–283, 2012.

[29] J. Li, Q. Wang, C. Wang, and K. Ren, "Enhancing attribute-based encryption with attribute hierarchy," *Mob. Netw. Appl.*, vol. 16, no. 5, pp. 553–561, 2011.

[30] A. Lewko and B. Waters, "Unbounded hibe and attribute-based encryption," in *Proc. Adv. Cryptol. (EUROCRYPT'11)*, 2011, pp. 547–567.

[31] N. Attrapadung, B. Libert, and E. de Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Public Key Cryptography (PKC'11)*, vol. 6571, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Berlin, Germany: Springer, 2011, pp. 90–108.

[32] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. Adv. Cryptol. Conf. (EUROCRYPT'11)*, vol. 6632, 2011, pp. 568–588.

[33] S. Yu, K. Ren, and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 673–686, Apr. 2011.

Chun-I Fan received the MS degree in computer science and information engineering from National Chiao Tung University, Taiwan, in 1993, and the PhD degree in electrical engineering at National Taiwan University in 1998. From 1999 to 2003, he was an associate researcher and project leader of Telecommunication Laboratories, Chunghwa Telecom Co., Ltd, Taiwan. In 2003, he joined the faculty of the department of computer science and engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, and has been a full professor since 2010. His current research interests include applied cryptology, cryptographic protocols, information and communication security, and he has published over 100 technical papers. He won the Dragon PhD Thesis Award from Acer Foundation, Best PhD Thesis Award from Institute of Information & Computing Machinery in 1999, Best Student Paper Awards in National Conference on Information Security 1998. He advised his graduate students to win the Best Student Paper Awards in National Conference on Information Security 2007, Best Master Thesis Award from Taiwan Association for Web Intelligence Consortium in 2011, Outstanding Master Dissertation Award from Taiwan Institute of Electrical and Electronic Engineering in 2011 and 2012, Master Thesis Award from Chinese Cryptology and Information Security Association in 2012, and Outstanding PhD Dissertation Award from Institute of Information & Computing Machinery in 2012. He also was the editor-in-chief of *Information Security Newsletter* and is an executive director of Chinese Cryptology and Information Security Association.

Vincent Shi-Ming Huang received the MS degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2006, and PhD degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2010. He now is an engineer in the Cloud Computing Center in Industry Technology Research Institute. His research interests include cloud computing and security, electronic payment, electronic commerce, network and communication security, information security, and applied cryptography.

He-Ming Ruan received the BS degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2006, and MS degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung Taiwan, in 2008. He is now a PhD candidate at the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. His research interests are primarily in information security.