Arbitrary-State Attribute-Based Encryption with Dynamic Membership

Speaker: Chun-I Fan

Chun-I Fan, National Sun Yat-sen University, Kaohsiung
Vincent Shi-Ming Huang, Industry Technology Research Institute, Hsinchu
He-Ming Rung, National Taiwan University, Taipei

*IEEE Transactions on Computers, 2014*
1. Introduction
2. Preliminaries
3. The Proposed Scheme
4. Security Proof
5. Comparisons
6. Conclusions
Public Key Infrastructure (PKI)

Traditional PKI has some features:

- A sender should get the public key (or the identity) of the receiver in advance.
- The more receivers the system has, the more bandwidth it consumes.
- Broadcast encryption may be able to solve the problem of performance, but the sender needs to have the receiver list.
In 1993, Fiat, and Naor proposed a broadcast encryption system.

In 2001, Boneh and Franklin proposed an identity-based encryption (IBE) scheme based on Weil pairing, and enhanced it by the technique from Fujisaki and Okamoto.

In 2005, Sahai and Waters proposed a new type of IBE, fuzzy IBE, which was the prototype of ABE.

In 2007, Baek, Susilo, and Zhou presented another fuzzy IBE with new construction.
Our Contribution

A melancholia patient

He wants to share the experiences of struggling against melancholia on a platform

with doctors

with other male patients

Attribute:

((“Gender: MALE” AND “MENTAL DISORDER: MELANCHOLIA”) OR (“CAREER: DOCTOR” AND “SPECIALITY: MELANCHOLIA”))

Our scheme is the first one which supports dynamic membership and arbitrary-state attributes.

It is CCA secure under a standard model (without using random oracles).
Dynamic Membership

- **Expandability.** A new user is able to enroll in the system.

- **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 are not required to interact with KGC (Key Generation Center) to renew their private keys.
Lagrange Interpolation

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

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

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

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

- **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 $G_T$.

- **Computability**: There is an efficient algorithm to compute $e(P, Q), \forall P \in G_0, Q \in G_1$. 

Let $G_0, G_1, \text{ and } G_T$ 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 $Z \in_R \{e(P, Q)^{abc}, Y \in_R G_T\}$, decide if $Z = e(P, Q)^{abc}$.
Definition 1 (The DBDH Assumption)

We define that an adversary $\mathcal{C}$ with an output $b' \in \{0, 1\}$ has advantage $\epsilon'$ 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 $Z \in \{e(P, Q)^{abc}, Y \in_R G_T\}$. 
An Access Tree

- **Node R**
  - Index: 1
  - Value: 2

- **Node C**
  - Index: 1
  - Value: 2

- **Node A**
  - Index: 1
  - Value: 2

- **Node B**
  - Index: 1
  - Value: 1

- **Node D**
  - Index: 1
  - Value: 1

**Attribute-Value Pairs**
- **A**
  - Attribute: Career
  - Value: Doctor

- **B**
  - Attribute: Speciality
  - Value: Melancholia
The Proposed Scheme
Overview: Enrollment, Encryption, Decryption

Setup: (PK, MK) → KGC: (Ai, (mj,j)∀j∈Ai, MK)

D_i and updated PK

User i: (Ti, M, PK) → Encryption

CT

Decryption: (CT, Dj) → User j: M

(CT, Dj') → User j': ⊥
The Proposed Scheme
Overview: Leaving, Updating

- User $u$ leaves the system.
- User $i$ updates an attribute $D'_i$.
- KGC generates $m'_{i,j}$.
- Updated key $(D'_i, PK')$ is sent to $u$.
- $u$ receives $PK'$ and updates its key.
The Proposed Scheme

**Setup**($k$)

1. **Step 1:** $G_0, G_1, G_T$ with prime order $q$, $e : G_0 \times G_1 \rightarrow G_T$, generator $P$ of $G_0$, generator $Q$ of $G_1$
2. **Step 2:** $\alpha, \beta \in_R \mathbb{Z}_q^*$ and $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$.
3. **Step 3:** Generate two default users:
   - $U = \{0, 1\}$, $\{v_{i,j}\}_{\forall i \in U, j \in A} \in_R \mathbb{Z}_q^*$, $\{t_i\}_{\forall i \in U} \in_R \mathbb{Z}_q^*$.
   - $V = (\prod_{\forall i \in U} v_{i,j})Q_{\forall j \in A}$
   - $\overline{v_{i,j}} = t_i \prod_{\forall k \neq i, k \in U} v_{k,j}^{-1} + v_{i,j} \mod q_{\forall i \in U, \forall j \in A}$.
4. **Step 4:** Public parameter $PK = (G_0, G_1, G_T, e, H, P, Q, U = e(P, Q)^{\alpha(\beta-1)}, e(P, Q)^{\alpha\beta}, \{V_j\}, \{\overline{v_{i,j}}\}_{\forall i \in U})_{\forall j \in A, \forall}$
   - The master key of KGC is $MK = \alpha Q$.

The asymmetric setting of $e$ can be implemented by **BN-Curve** which is with faster software implementations.
The Proposed Scheme
Enrollment($A_i, (m_{i,j})_{\forall j \in A_i}, MK$)

- **Step 1:** $r_i, t_i, \{v_{i,j}\}_{\forall j \in A_i}, \{r_{i,j}\}_{\forall j \in A_i} \in R \mathbb{Z}_q^*$
- **Step 2:** $U = U \cup \{i\}$
- **Step 3:**
  \[
  \begin{align*}
  \{h_{i,j} = H(m_{i,j})\}_{\forall j \in A_i} \\
  \{V_j = v_{i,j} V_j\}_{\forall j \in A} \\
  \{v_{i,j} = t_i \prod_{\forall k \neq i, k \in U} v_{k,j}^{-1} + v_{i,j} \text{ mod } q\}_{\forall j \in A} \\
  \{v_{k,j} = (v_{k,j} - v_{k,j}) v_{i,j}^{-1} + v_{k,j} \text{ mod } q\}_{\forall \forall k \neq i, k \in U, \forall j \in A}
  \end{align*}
  \]
- **Step 4:** Generate user $i$'s private key $D_i =$
  \[
  \begin{align*}
  D_i &= \alpha Q + t_i r_i Q \\
  \{D_{i,j} = v_{i,j}^{-1} (r_i P + r_{i,j} h_{i,j} P)\}_{\forall j \in A_i} \\
  \{D'_{i,j} = t_i r_{i,j} P\}_{\forall j \in A_i} \\
  \{D''_{i,j} = r_i P + r_{i,j} h_{i,j} P\}_{\forall j \in A_i}
  \end{align*}
  \]
- **Step 5:** Increase $V$ and update ($\{V_j, \{\overline{v_{i,j}}\}_{\forall i \in U}\}_{\forall j \in A}$, $V$) in $PK$. 
KGC increases $\mathcal{V}$ and updates $PK$ as follows:

$$\begin{align*}
\{V_j = v_{u,j}^{-1} V_j\} & \forall j \in A \\
\{\overline{v_{k,j}} = (\overline{v_{k,j}} - v_{k,j}) v_{u,j} + v_{k,j} \mod q\} & \forall k \neq u, k \in \mathcal{U}, \forall j \in A
\end{align*}$$

Finally, it sets $\mathcal{U} = \mathcal{U} \setminus \{u\}$ and deletes $\{\overline{v_{u,j}}\} \forall j \in A$ in $PK$. 
The Proposed Scheme

Updating $\left(m'_{i,j}\right)$

- **Step 1:** $v'_{i,j}$, $r'_{i}$, $r'_{i,j} \in \mathbb{Z}_q^*$
- **Step 2:** $h'_{i,j} = H(m'_{i,j})$
- **Step 3:** Give

\[
\begin{aligned}
D_i &= \alpha Q + t_i r'_i Q \\
D_{i,j} &= v'_{i,j}^{-1} (r'_i P + r'_{i,j} h'_{i,j} P) \\
\{D_{i,k} &= v'_{i,k}^{-1} (r'_i P + r_{i,k} h_{i,k} P)\} \forall k \in A_i \setminus \{j\} \\
D'_{i,j} &= t_i r'_{i,j} P \\
D''_{i,j} &= r'_i P + r'_{i,j} h'_{i,j} P \\
\{D''_{i,k} &= r'_i P + r_{i,k} h_{i,k} P\} \forall k \in A_i \setminus \{j\}
\end{aligned}
\]

...to user $i$.

- **Step 4:** Increase $\nabla$ and update $PK$.

\[
\begin{aligned}
V_j &= v'_{i,j} v'_{i,j} V_j \\
\overline{v}_{i,j} &= (\overline{v}_{i,j} - v_{i,j}) + v'_{i,j} \mod q \\
\overline{v}_{k,j} &= (\overline{v}_{k,j} - v_{k,j}) v_{i,j} v'_{i,j}^{-1} + v_{k,j} \mod q, \forall k \in \mathcal{U} \setminus \{i\}
\end{aligned}
\]
Access tree structure construction.

- **For the root node** $R$:
  1. $s \in R \mathbb{Z}_q^*$ and $k_R$ is the threshold value of $R$.
  2. Randomly choose a polynomial $q_R$ of degree $d_R = k_R - 1$ with $q_R(0) = s$.
  3. Assign a unique index number $x$ for each child of $R$.

- **For each internal node** $N$ other than $R$:
  1. $k_N$ is the threshold value of $N$.
  2. Randomly choose a polynomial $q_N$ of degree $d_N = k_N - 1$ with $q_N(0) = q_{\text{parent}(N)}(\text{index}(N))$.
  3. 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 $q_{N_L}(0) = q_{\text{parent}(N_L)}(\text{index}(N_L))$. 

**Ciphertext generation:** \( K \in_R G_T \) and \( \mathcal{N}_L = \{ \text{the leaves of } T \} \).

The ciphertext is

\[
CT = (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_{\text{att}(N)}, C'_N = q_N(0)H(\text{val}(N))Q, \\
\{ \overline{v_{i,\text{att}(N)}} \}_i \forall i \in \mathcal{U} \} \forall N \in \mathcal{N}_L, \forall V \).
\]

\( \{ \{ \overline{v_{i,\text{att}(N)}} \}_i \forall i \in \mathcal{U} \} \forall N \in \mathcal{N}_L \) can be excluded from the ciphertext.
DecryptNode(\(CT, D_i, N\)): Let \(V_j = v_j P\).

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

\[
\text{DecryptNode}(\!CT, \!D_i, \!N\!)
= \frac{e(D_{i,j}, \overline{v_{i,j}C_N})}{e(D'_{i,j}, C'_{N}) e(D''_{i,j}, C_N)}
= \frac{e(v_{i,j}^{-1}(r_i P + r_{i,j} h_{i,j} P), (t_i v_{j}^{-1} v_{i,j} + v_{i,j}) q_N(0) v_j Q)}{e(t_i r_{i,j} P, q_N(0) h_{i,j} Q) e(r_i P + r_{i,j} h_{i,j} P, v_j q_N(0) Q)}
= \frac{e(r_i P + r_{i,j} h_{i,j} P, t_i v_{i,j} q_N(0) Q + v_{i,j} v_j q_N(0) Q)}{e(r_{i,j} P, t_i q_N(0) h_{i,j} Q)}
= e(P, Q)^{t_i r_{i,j} q_N(0)}.
\]

Otherwise, \(\text{DecryptNode}(CT, D_i, N) = \bot\).
If $N$ is an internal node:

1. For each child $N_c$ of $N$, $F_{N_c} = DecryptNode(CT, D_i, N_c)$.

2. Let $I_c$ be a $k_N$-sized set containing the indexes of the child nodes $N_c$'s such that $F_{N_c} \neq \bot$ for each $N_c$. Return

$$F_N = \prod_{\forall \text{index}(N_c)=z \in I_c} F_{N_c}^{\Delta z, I_c(0)}$$

$$= \prod_{\forall \text{index}(N_c)=z \in I_c} (e(P, Q)^{t_ir_iq_{N_c}(0)})^{\Delta z, I_c(0)}$$

$$= \prod_{\forall \text{index}(N_c)=z \in I_c} (e(P, Q)^{t_ir_iq_{\text{parent}(N_c)}(z)})^{\Delta z, I_c(0)}$$

$$= \prod_{\forall \text{index}(N_c)=z \in I_c} (e(P, Q)^{t_ir_iq_{N}(z)})^{\Delta z, I_c(0)}$$

$$= e(P, Q)^{t_ir_iq_{N}(0)}$$

3. If no such set exists, $N$ is not satisfied and return $F_N = \bot$. 


The Proposed Scheme
Decryption

Call $DecryptNode(CT, D_i, R)$:

$$A = DecryptNode(CT, D_i, R)$$

$$= e(P, Q)^{t_ir_iqR(0)}$$

$$= e(P, Q)^{t_ir_is}$$

Compute the session key $K$:

$$\frac{A \cdot \tilde{C}}{e(C, D_i) \cdot C'} = \frac{e(P, Q)^{t_ir_is} \cdot e(P, Q)^{\alpha\beta S} K}{e(sP, (\alpha + t_ir_i)Q) \cdot e(P, Q)^{\alpha(\beta - 1)s}}$$

$$= \frac{e(P, Q)^{\alpha\beta S} \cdot e(P, Q)^{t_ir_is} K}{e(P, Q)^{(\alpha S + \alpha(\beta - 1)s + t_ir_is)}}$$

$$= \frac{e(P, Q)^{\alpha\beta S + t_ir_is} K}{e(P, Q)^{\alpha\beta S + t_ir_is}}$$

$$= K$$

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

- The *Enrollment* algorithm:
  1. Expandability

- The *Leaving* and *Updating* algorithms:
  1. Revocability
  2. Renewability
  3. Independence
The proposed CP-ABE-DM scheme is $C_{CA_{DM}}$ secure under the DBDH assumption in a standard model.
Security Proof

The $CCA_{DM}$ Game

\[ PK = (G_0, G_1, G_T, e, H, P, Q, U = e(aP, bQ - Q), e(aP, bQ)) \]
\[ MK = aQ \]

The Decisional Bilinear Diffie-Hellman Problem
\((G_0, G_1, G_T, q, P, Q, aP, bP, cP, aQ, bQ, cQ, Z)\)

\[ A \]
\[ PK \]
Queries / Responses
\((M_0, M_1, T^*) \) or
\((M_0, M_1, T^*, i, j, m^*) \)
\((T^*, Z \cdot K, cP, e(aP, eQ)), M = E_K(M_{b''}), H(K\|M)P, \{C_N, C'_N, \{v_i, att(N)\}_\forall i \in U\}_\forall N \in N_L, V) \]
Queries / Responses
\[ \]A outputs \(b''' \in \{0, 1\} \)
\[ C \]
\textbf{Setup}
\textbf{Enrollment}(\(S_i\))
\textbf{Leaving}(i) \(b'' \in \mathbb{R} \{0, 1\} \)
\textbf{Updating}(i, j, m)
\textbf{Decryption}'(CT)

\[ C \] solves the Decisional Bilinear Diffie-Hellman problem with non-negligible advantage.

If \(b''' = b'', \) \(C\) outputs \(b' = 1\), otherwise \(b' = 0\)
Security Proof

The Cipher Algorithm

\[ \text{Cipher}(\mathcal{T}, N, \mathcal{C}_0) \]

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

Initially, \( \mathcal{C} \) calls \( \text{Cipher}((\mathcal{T}^*), R, cQ) \).
Comparisons

- **Dynamic Membership**: It allows an ABE system to manage member enrollment, attribute updating, and member revocation efficiently.

- **Sender Updating**: A sender must grab the newest public information before she/he encrypts a message.

- **Receiver Updating**: A receiver must interact with the system to refresh her/his private key or retrieve the newest public information before she/he decrypts a ciphertext.

- **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.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>Dynamic Membership</th>
<th>ASA</th>
<th>Special Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Updating</td>
<td>Leaving</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[4]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[5]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>[12]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[13]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[15]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[7]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[8]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[9]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[10]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Scheme</td>
<td>Dynamic Membership</td>
<td>ASA</td>
<td>Special Feature</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------</td>
<td>-----</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>Updating</td>
<td>Leaving</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>[11]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[14]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[25]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[26]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[27]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[28]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[29]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[30]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[31]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[32]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[33]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

ASA: Arbitrary-State Attribute
<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>the number of the members in an ABE system</td>
</tr>
<tr>
<td>( m )</td>
<td>the number of the attributes provided in an ABE system</td>
</tr>
<tr>
<td>( m_c )</td>
<td>the number of attributes associated to a ciphertext</td>
</tr>
<tr>
<td>( m_d )</td>
<td>the maximum of (</td>
</tr>
<tr>
<td>( m_u )</td>
<td>the number of a user's attributes</td>
</tr>
<tr>
<td>( m_a )</td>
<td>the number of authorities</td>
</tr>
<tr>
<td>( m_{or} )</td>
<td>the number of OR gates of the access rule associated to a ciphertext</td>
</tr>
<tr>
<td>( m_{and} )</td>
<td>the number of the AND conjunction attributes of the access rule associated to a ciphertext</td>
</tr>
<tr>
<td>( m_{not} )</td>
<td>the number of the NOT conjunction attributes of the access rule associated to a ciphertext</td>
</tr>
<tr>
<td>DR, IR</td>
<td>Direct Mode and Indirect Mode</td>
</tr>
<tr>
<td>SA, OA</td>
<td>Subject Attribute and Objected Attribute</td>
</tr>
</tbody>
</table>
## Performance Comparisons: Computation Cost

<table>
<thead>
<tr>
<th></th>
<th>Encryption Cost of the Sender</th>
<th>Decryption Cost of the Receiver</th>
<th>The Necessary Computation Cost of the Center</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ours</strong></td>
<td>$O(m_c)$</td>
<td>$O(m_c)$</td>
<td>$O(m + m_u)$</td>
</tr>
<tr>
<td>[4]</td>
<td>$O(m \times m_c + m_d)$</td>
<td>$O(m_c)$</td>
<td>$O(m \times m_P \times m_u \times m)$</td>
</tr>
<tr>
<td>[5]</td>
<td>$O(m)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m \times m_u)$</td>
</tr>
<tr>
<td>[12]</td>
<td>$O(m \times m_c)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m \times m_u)$</td>
</tr>
<tr>
<td>[13]</td>
<td>$O(m_{and} \times m_{or})$</td>
<td>$O(m_c)$</td>
<td>$O(m_u)$</td>
</tr>
<tr>
<td>[15]</td>
<td>$O(m_c)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m^2)$</td>
</tr>
<tr>
<td>[7]</td>
<td>$O(m_c)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[8]</td>
<td>$O(m_c)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m)$</td>
</tr>
<tr>
<td>[9]</td>
<td>$O(m_c)$</td>
<td>$O(m_c \times m_L)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>[10]</td>
<td>$O(m_{and} + m_{not})$</td>
<td>$O(m_u + m_{not})$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[11]</td>
<td>$O(m_c)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[14]</td>
<td>$O(m \times m_c + m_c^2)$</td>
<td>$O(m \times m_c m_c^2)$</td>
<td>$O(n \times m_u^2)$</td>
</tr>
<tr>
<td>[25]</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[26]</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[27]</td>
<td>$O(m)$</td>
<td>$O(m_c \times m_u)$</td>
<td>$O(n \times m_a \times m_u)$</td>
</tr>
<tr>
<td>[28]</td>
<td>$O(m_a + m_c)$</td>
<td>$O(m_a \times m_u)$</td>
<td>$O(n \times m_a \times m_u)$</td>
</tr>
<tr>
<td>[29]</td>
<td>$O(m_c^2)$</td>
<td>$O(m_a^2)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[30]</td>
<td>$O(m_c)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[31]</td>
<td>$O(m)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[32]</td>
<td>$O(m_c)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[33]</td>
<td>$O(m_c)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
</tbody>
</table>
## Performance Comparisons: Storage and Communication Cost

<table>
<thead>
<tr>
<th>Ours</th>
<th>Size of Private Key</th>
<th>Size of Ciphertext</th>
<th>Size of Public Parameters</th>
<th>Enrollment (between the center and a user)</th>
<th>Updating (between the center and a user)</th>
<th>Leaving (between the center and a user)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(n \times m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u \times m_p)$</td>
<td>DR: $O(1)$, IR: $O(m_d)$</td>
</tr>
<tr>
<td>[4]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(n + m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[5]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(n + m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[12]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(n + m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[13]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(n + m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[15]</td>
<td>$O(m)$</td>
<td>$O(m_c)$</td>
<td>$O(m^2)$</td>
<td>$O(m^2)$</td>
<td>$O(n \times m^2)$</td>
<td>$O(n \times m^2)$</td>
</tr>
<tr>
<td>[7]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[8]</td>
<td>$O(m)$</td>
<td>$O(m_c)$</td>
<td>$O(1)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m)$</td>
<td>$O(n \times m)$</td>
</tr>
<tr>
<td>[9]</td>
<td>$O(m)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[10]</td>
<td>$O(m_a)$</td>
<td>$O(m_a + m_c)$</td>
<td>$O(m_a)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[11]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[14]</td>
<td>$O(m_a \times m_u)$</td>
<td>$O(m_a \times m_c)$</td>
<td>$O(m_a)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[25]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[26]</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[27]</td>
<td>$O(m_a \times m_u)$</td>
<td>$O(m_a \times m_c)$</td>
<td>$O(m_a)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[28]</td>
<td>$O(m_a \times m_u)$</td>
<td>$O(m_a \times m_c)$</td>
<td>$O(n \times m_a)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[29]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[30]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(1)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[31]</td>
<td>$O(m \times m_u)$</td>
<td>$O(1)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[32]</td>
<td>$O(m_u)$</td>
<td>$O(m_c)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
<tr>
<td>[33]</td>
<td>$O(m_u)$</td>
<td>$O(m)$</td>
<td>$O(m)$</td>
<td>$O(m_u)$</td>
<td>$O(n \times m_u)$</td>
<td>$O(n \times m_u)$</td>
</tr>
</tbody>
</table>
Conclusions

- An attribute-based encryption scheme with **dynamic membership** has been proposed.

- This is the first ABE scheme which can support **arbitrary-state attributes** and attribute (and value) updating with **Sender Updating** only.

- It has been formally proved to be **CCA secure** under a **standard model**.
Thanks for Listening