Multi-authority Attribute-Based Encryption with User Revocation and Outsourcing Decryption

Xinyue Zhang¹, Yue Chen¹, Xincheng Yan¹ and Hongyong Jia²

¹PLA Information Engineering University, Zhengzhou 450001, China
²School of Software and Applied Technology, Zhengzhou University, Zhengzhou 450001, China
Email: zwuyue@163.com

Abstract. Attribute based encryption scheme is widely used to share sensitive data in cloud storage environment because it can realize fine-grained access control. Aiming at the problem of user revocation and high cost of decryption in cloud storage environment, a multi-authority attribute based encryption scheme with efficient user revocation and outsourcing decryption was proposed. It supports large universe. Each authority manages a specific set of attributes and distributes attribute keys to users with corresponding attributes, avoiding the problem that a single authority may easily become a system bottleneck. The user revocation algorithm of the scheme can realize the efficient revocation of the authorized users, ensuring the forward security of ciphertext. It’s proved to be statically-secure under random oracle and the efficiency analysis shows that it reduces user decryption cost greatly.

1. Introduction
With the rapid development of cloud storage technology, more and more users tend to outsource data to cloud storage. However, it’s a big problem to ensure user privacy and data security when outsourcing sensitive data in a semi-trusted cloud storage environment. The attribute-based encryption (ABE) proposed by Sahai et al. [1] can protect data security and implement fine-grained access control. In ciphertext-policy attribute based encryption (CPABE) [2], encipherers can develop their own access control policies, which make it suitable for cloud storage environments.

In the basic ABE, attributes are managed by a single authority. However, it may easily become a system bottleneck. To solve this problem, the multi-authority ABE was proposed (MA-ABE) [3]. And many solutions solved the problem of user revocation in MA-ABE [4-5]. However, many complex exponential operations and bilinear operations should be computed in these schemes and it’s costly for some users with weak computing power.

In the single attribute authority construction, once the attribute authority is corrupted, the security will be completely destroyed. In order to increase the robustness, many MA-ABE constructions were presented in recent years. The disadvantage in [6] is that each authority needs to be identified and interact with each other during system setup, which is not convenient for expansion. Rouselakis et al [7] presented a MA-ABE supporting large universe. The constructions in [8, 9] realized user privacy protection by dividing attributes and users into different domains in [8] and anonymous key distribution protocol in [9]. A MA-ABE construction for cloud computing environment was presented by Sushmita et al. [10] and a construction with user revocation was given in [11]. However, none of the above solutions support outsourcing decryption. For users with limited computing ability, the decryption overhead is expensive. A construction with outsourcing decryption was given by Zhang et al.in [12] but the computation of user decryption process is still related to the complexity of the access structure. Thus,
a construction with user revocation, outsourcing decryption and large universe was proposed. User
decryption process requires only one exponential operation and it’s suitable for data access control in
cloud storage environment.

2. Background

2.1. Linear Secret Sharing Schemes

Definition 2 (Linear Secret Sharing Scheme (LSSS) [13]) A secret sharing scheme $\Pi$ over a set of
parties $\mathcal{P}$ is called linear if

(1) The shares for each party form a vector over $\mathbb{Z}_p$.

(2) There exists a matrix $M$ with $n$ rows and $l$ columns called the share-generating matrix for $\Pi$ and
$(M, \rho)$ represents an access structure $\mathcal{A}$. For all $x = 1, \ldots, n$, $\rho$ is a mapping from $\{1, \ldots, n\}$ to $\mathcal{P}$ such
that the $x$-th line of matrix $M$ is mapped to one participant $P_i$. A column vector $v = (s, r_1, \ldots, r_l)$, where
$s \in \mathbb{Z}_p$ is the secret to be shared and $r_1, \ldots, r_l \in \mathbb{Z}_p^\ast$ are randomly chosen, is $l$ dimensional. Then $Mv$ is
the vector of $n$ shares of secret $s$ according to $\Pi$. The share $\lambda_i = (Mv)_i$ belongs to party $\rho(x)$.

According to [13], each secret sharing scheme should satisfy the reconstruction requirement. Let $S$ denote
an authorized set of parties and let $\Gamma$ be the set of rows whose labels are in $S$, i.e.
$\Gamma = \{x | \rho(x) \in S\}$. There exist constants $\{c_i \in \mathbb{Z}_p\}_{i \in \Gamma}$ in $\mathbb{Z}_p$ such that for any valid shares $\lambda_i = (Mv)_i$ of
a secret according to $\Pi$, it is true that: $\sum_{i \in \Gamma} c_i \lambda_i = s$, or equivalently $\sum_{i \in \Gamma} c_i M_{is} = (1, 0, \ldots, 0)$, where
$M_{is}$ is the $x$-th row of $M$. For unauthorized sets, such constants do not exist.

If the access structure is encoded as a monotonic Boolean formula, a generic algorithm that generates
the corresponding access policy in polynomial time was given in [3].

2.2. Bilinear Maps

Definition 3 (Bilinear Maps) Let $G_i$ and $G_r$ be two multiplicative cyclic groups of prime order $p$. Let
g denote a generator of $G_i$. $e : G_i \times G_i \rightarrow G_r$ is a bilinear map with the following properties:

(1) Bilinear: for all $u, v \in G_i$ and $a, b \in \mathbb{Z}_p$, $e(a^u, b^v) = e(a, v)^{ab}$;

(2) Non-degenerate: $e(g, g) \neq 1$;

(3) Computable: for all $u, v \in G_i$, the bilinear map $e(u, v)$ is computable in polynomial time.

2.3. Complexity Assumption

Definition 4 (Modified q-Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption (q-
DPBDH-E2) [7]) Choose a bilinear group $G_i$ of order $p$ according to the security parameter $\mathcal{K}$, which
admits a non-degenerate bilinear mapping $e : G_i \times G_i \rightarrow G_r$. Pick $s, a, b_1, b_2, \ldots, b_q \leftarrow \mathbb{Z}_p$ and $R \leftarrow G_r$ randomly. Let

$$D = \left( G, p, e, g, g', \left\{ g^{\alpha_i j} \right\}_{i \in [2q]} \right) \left( g^{b_i} \right)_{i \in [2q]}, \left( g^{j \beta_i} \right)_{i \in [2q]}, \left( g^{\alpha_i j \beta_i} \right)_{i \in [q]} \left( g^{a j \beta_i} \right)_{i \in [q]} \left( g^{a j \beta_i} \right)_{i \in [q]} \left( g^{a j \beta_i} \right)_{i \in [q]}$$ \hspace{1cm} (2.1)

The assumption states that no polynomial-time distinguisher can distinguish the distribution
$(D, e(g, g)^{\alpha_\cdot \cdot})$ from the distribution $(D, R)$ with more than negligible advantage.
3. UR-OD-MA-CPABE

3.1. Scheme Model

A multi-authority CP-ABE access control scheme with efficient user revocation and outsourced decryption (UR-OD-MA-CPABE) was proposed based on the construction in [7]. The scheme model is shown in Figure 1. It mainly consists of the following entities: data owner, cloud server (CS), central authority (CA), attribute authority (AA) and users. CA is responsible for user registration, generation of the global parameters and the master secret key MSK. Each AA manages a set of attributes and issues attribute secret keys to users. The attribute sets managed by different AAs are disjoint. CS stores the encrypted data of the data owner, does partial decryption and updates ciphertext. CS is assumed to be semi-trusted and will not collude with the unauthorized or revoked users. The data owner formulates an access structure, encrypts the data and uploads it to the CS. A user with his secret key can access to the encrypted data in cloud.

![Scheme model](image)

Figure 1. Scheme model.

3.2. Static Security

We define the static security game between a challenger and an adversary in this section. To simulate a collusion attack by multiple users, the adversary is allowed to query the attribute secret key of multiple users, and the adversary is also allowed to query the secret keys of multiple users who do not satisfy the access policy. We also allow the adversary to corrupt a certain set of AAs and he can generate their public and secret keys. The security game between the adversary and the challenger is described as follows.

(1) System Setup: The challenger calls SystemSetup(l*) \( \rightarrow \) GP and gives GP to the adversary.

(2) Adversary’s Queries: The adversary submits the challenged access structure \( \mathcal{A} \), two messages \( m_0, m_1 \) of equal length, a set of GIDs and their attributes \( \{(GID_i, S_i)\}_i \) to the challenger. The adversary also declares a set \( \mathcal{C}_{\mathcal{A}_i} \) of corrupted AAs that he can create in a malicious way. Let \( S_{\mathcal{C}_{\mathcal{A}_i}} \) represent the set of attributes managed by AA in \( \mathcal{C}_{\mathcal{A}_i} \) and \( \mathcal{N}_{\mathcal{A}_i} \) represent the set of non-corrupted AAs. The adversary can make the following queries:

- The \( PK_{\mathcal{A}_i} \) of a non-corrupted \( \mathcal{A}_i \)
- Attribute secret keys \( \{SK_{(GID_i, S_i)}\}_{u \in S_0} \) of a user with \( (GID_i, S_i) \)
- The public key and secret key \( \{PK_{GID_i}, SK_{GID_i}\} \) of a user with \( (GID_i, S_i) \), meeting the requirement \( S \cap S_{\mathcal{C}_{\mathcal{A}_i}} \) doesn’t satisfy the access structure
(3) Challenger’s Replies: The challenger flips a random coin \( b \in \mathbb{Z}_2 \{0,1\} \) and replies the queries together with ciphertext \( CT^* \) of \( m_b \) under the access policy \( A \).

### 4. System Design

#### 4.1. System Setup
- CA setup: CA generates global parameters and the master secret key. Let \( G_t \) and \( G_r \) be two cyclic groups of prime order \( p \), \( g \) be a generator of \( G_t \), and \( e : G_t \times G_t \rightarrow G_r \) be a bilinear map. \( H_t : \mathcal{U}_{gid} \rightarrow G_t, \ H_r : \mathcal{U}_{gid} \rightarrow G_r, \ F : \mathcal{U}_{gid} \rightarrow \mathbb{Z}_p \) are three hash functions. \( T \) is a computable function mapping a attribute to the related AA. Pick \( g_i \in G_t \), \( y \in \mathbb{Z}_p \) randomly. Let \( GP = \{ G_t, G_r, p, g, e, H_t, H_r, F, T, g_i \} \). MSK = \( y \), public key MPK = \( g^y \).
- AA setup: Each AA picks \( \alpha, \beta, \gamma \in \mathbb{Z}_p \) randomly, and generates the secret key \( SK _ i = \{ \alpha, \beta, \gamma \} \), the public key \( PK _ i = \{ e(g, g)^{\alpha}, g^\beta, g^\gamma \} \).

#### 4.2. User Key Generation
- CA assigns a unique identity GID to the user and selects \( u_{gid} \in \mathbb{Z}_p \). The user’s secret key \( SK_{gid} = u_{gid}^{-1} \), the user’s public key \( PK_{gid} = g^{u_{gid}} \), update key \( SK'_{gid} = g_1^{f(gid)(f(gid)+y)} \).
- For each attribute \( u \in S_{gid} \), the related AA selects \( \epsilon \in \mathbb{Z}_p \), and the attribute secret key is \( SK_{gid,a} = \{ K_{gid,a} = g^{u_{a\text{a}}} H_i(gid)^{\beta} H_j(u)^{g_1^{f(gid)(f(gid)+y)}} \} \).

#### 4.3. Encrypt
The data owner defines an LSSS for a message \( K \) and the access structure is represented as \( A = (M, \rho) \). \( M \) is a \( n \times l \) access matrix and \( \rho \) is a function mapping the rows of matrix \( M \) to the attributes. Let \( \delta(x) = T(\rho(x)) \) map some row \( M_x \) to the related AA. The data owner chooses vectors \( v = (v_1, v_2, v_3, \ldots, v_l) \) and \( w = (0, w_1, \ldots, w_l) \), where \( v_1, v_2, v_3, \ldots, v_l, w_1, \ldots, w_l \in \mathbb{Z}_p \). Let \( \lambda_i \) denote the share of \( v \) corresponding to row \( M_x \), i.e. \( \lambda_i = M_x \cdot v \) and let \( \omega_i \) denote the share of \( 0 \), i.e. \( \omega_i = M_x \cdot w \). For each row \( M_x \), a random \( \eta \in \mathbb{Z}_p \) is picked, the ciphertext is calculated as

\[
CT = \{ \delta, c_0 = K \cdot e(g, g), (C_{i_1})_{\gamma} = e(g, g)^{\omega_i}, C_{i_2} = g^{\lambda_i}, C_{i_3} = g^{\omega_i \cdot \lambda_i}, C_{i_4} = (H_j(\rho(x)))_{\gamma_4}, C_{i_5} = g^{\omega_i \cdot \lambda_i} \}_{i=0}^l
\]

In practice, the message \( K \) can be a symmetric key and the files can be encrypted under it.

#### 4.4. Decrypt
- Outsourcing decryption: a user with limited computation ability can upload his attribute secret key \( \{ SK_{gid,a} \}_{a \in S_{gid}} \) to cloud server. If \( S_{gid} \) satisfy the access policy \( A \), there are \( \{ SK_{gid,p(a)} \} \) for a subset of rows \( M_x \) such that (1,0,...,0) is in the span of these rows and let \( \Gamma \) be the index of these rows. Therefore, there exist constants \( c_i \in \mathbb{Z}_p \) such that \( \sum_{i \in \Gamma} c_i M_x = (1,0,\ldots,0) \), i.e. \( \sum_{i \in \Gamma} c_i \lambda_i = v \) and \( \sum_{i \in \Gamma} c_i \omega_i = 0 \). The cloud server computes

\[
C_{1,gid} = \prod_{i \in \Gamma} (C_{i_1})_{\gamma_i} = e(g, g)^{\omega_i} \cdot \prod_{i \in \Gamma} e(g, g)^{\delta_i^\rho \delta_i^\rho}_{\rho(a)}
\]

\[
C_{2,gid} = \prod_{i \in \Gamma} e(K_{gid,p(a)}, C_{i_2}) \cdot e(H_j(gid), C_{i_3}) \cdot e(C_{i_4}, K_{gid,p(a)}) \cdot e(g_1^{f(gid)}, C_{i_5})_{\gamma_i} \cdot \prod_{i \in \Gamma} e(g, g)^{\omega_i \cdot \lambda_i}_{\rho(a)}
\]

In practice, the message \( K \) can be a symmetric key and the files can be encrypted under it.
• CS sends the partially decrypted ciphertext $C_{GID} = \{C_0, C_{1,GID}, C_{2,GID}\}$ to the user GID, and according to his user secret key $SK_{GID} = u_{GID}^{-1}$, the user computes $K = \frac{C_0}{C_{1,GID} (C_{2,GID})^{u_{GID}^{-1}}}$.

Our construction supports outsourcing decryption and outsource complex bilinear operations and exponential operations to CS, reducing the computational cost of users. Since CS does not have the user secret key, the plaintext cannot be calculated.

### 4.5. User Revocation

• When user revocation occurs, CA writes the revoked users’ GID to the revocation list RL, and CS performs re-encryption to update the ciphertext so that the revoked users cannot continue decrypting the ciphertext. CS chooses a random vector $j = (j_1, j_2, \ldots, j_r)$ where $j_1, j_2, \ldots, j_r \in \mathbb{Z}_p$ and calculates the shares $\mu_k = M_s \cdot j$. Assuming the number of revoked users is $r$, it chooses $z_1, \ldots, z_r \leftarrow \mathbb{Z}_p$ such that $\sum_{k=1}^{r} z_k = z$ and compute

$$
\{C_{k,x} = g^{r_{k,x}} \cdot g^{\mu_k}\}_{x \in \{x_{\mathbb{Z}_p}\}} \tag{4.4}
$$

$$
\{D_{1,k} = (g^x g^{F(GID)})^{-\lambda_k}, D_{2,k} = g^\alpha\}_{x \in \{x_{\mathbb{Z}_p}\}} \tag{4.5}
$$

The final updated ciphertext is $CT' = \{A, C_{GID}, C_{1,GID}, C_{2,GID}, C_{3,GID}, C_{4,GID}\}_{x \in \{x_{\mathbb{Z}_p}\}} \{D_{1,k}, D_{2,k}\}_{x \in \{x_{\mathbb{Z}_p}\}}$.

• Unrevoked user decryption. For an unrevoked user GID, if $S_{GID}$ satisfy the access policy $\mathbb{A}$, there exist constants $c_s \in \mathbb{Z}_p$ such that $\sum_{s=1}^{r} c_s M_s = (1, 0, \ldots, 0)$, i.e. $\sum_{s=1}^{r} c_s \lambda_s = v \sum_{s=1}^{r} c_s \alpha_s = 0$ and $\sum_{s=1}^{r} c_s \mu_s = z$. User GID provides his update key $SK'_{GID} = g_1^{F(GID)} (F(GID)^{1+y})$ and CS computes

$$
C_{2,GID} = \prod_{s=1}^{r} (e(K_{GID,p(s)}, C_{2,s}) \cdot e(H_1(GID), C_{3,s}) \cdot e(C_{4,s}, K'_{GID,p(s)} \cdot e(g_1^{F(GID)}, C_{5,s})^x)) \tag{4.6}
$$

$$
D = \prod_{k=1}^{r} (e(SK_{GID}, D_{2,k}) \cdot e(g_1^{F(GID)}, D_{1,k}))^{y(F(GID)^{F(GID)+1})} = e(g_1, g)^{x F(GID)} \tag{4.7}
$$

$$
C'_{GID} = C_{2,GID} / D = \prod_{s=1}^{r} e(g, g)^{-y_{GID,p(s)} v_{c_s}} \tag{4.8}
$$

• CS sends $C_{GID} = \{C_0, C_{1,GID}, C_{2,GID}\}$ to the unrevoked user and he computes $K = \frac{C_0}{C_{1,GID} (C_{2,GID})^{u_{GID}^{-1}}}$.

### 5. Security Analysis

#### 5.1. Proof of Security

**Theorem 1** If the q-DPBDHE2 assumption holds, then for a challenge matrix of size at most $q \times q$ our scheme is statically secure in the random oracle model.

To prove the above theorem, the following lemmas are given.

**Lemma 1** Let $M \in \mathbb{Z}_p^{r \times n}$ be the secret sharing matrix of a LSSS for an access policy $\mathbb{A}$ and let $C \subseteq [n]$ be an unauthorized set of rows. Let $c \in \mathbb{N}$ be the dimension of the subspace spanned by the rows of $C$. There is a matrix $M^*$, where $M^*_{i,j} = 0$ for all $(x, j) \in C \times [l - c]$, satisfying $M^* \cdot v = M \cdot v$. In equation (5.1), the underlined rows consist the unauthorized set.
Lemma 2 If a user has attribute key $SK_{GID,u} = \{K_{GID,u}, K'_{GID,u}\}$, he can get a new key for $(GID,u)$ by picking $t' \in \mathbb{Z}_p$ and calculating $\{K_{GID,u}, H_2(u)\}$. For the ciphertext, the re-randomization can be done by picking a new $v' \in \mathbb{Z}_p$, new random vectors $v = (v_1',...,v_l')$ and $w = (w_1',...,w_l')$ and for each row $M_x$, choosing $r_x \in \mathbb{Z}_p$. The re-randomized ciphertext is

$$CT' = (A,C,e(g,g)^{v'} \cdot (C_x,e(g,g)^{w_1} \cdot e(g,g)^{q_{\gamma_1}} \cdot C_{\gamma_1},H_2(p(x)v') \cdot C_{\gamma_1},g^{\theta_{\gamma_1}}), C_x,g^{\theta_{\gamma_1}})_{\text{rand}}$$

Proof The static security game between a adversary $A$, a challenger $C$ and a simulator $B$ is as follows.

- $C$ provides $B$ with the q-DBDHE2 tuple $D$, where $Pr(T = e(g,g)^{\omega}) = Pr(T = R) = 1/2$.
- **Initiation** $B$ generates the global parameters $GP = \{G_1,G_2,e,p,g,g_1 = g^e,T\} \in \mathbb{Z}_p$ and sends to $A$.
- **Query Phase** $A$ should do all queries immediately after seeing the public parameters. $A$ sends the challenged policy $(M,\rho)$, two messages $m_0,m_1$ with equal length, $\{(GID_x,S_x)\}_{x=1}^n$ and the set of corrupted AAs $C_{AA}$ to $B$. Let $S_{\text{AA}}$ be the attribute set managed by AA in $C_{AA}$ and $\mathcal{I}_{\text{AA}}$ be the set of all the AAs.

According to lemma 1, $B$ transforms $M$ to $M'$ and let $l' = l - c$.

- AA’s PK: $B$ should provide $A$ with the PKs of the non-corrupted AAs as follows:
  1. If $AA \notin \mathcal{I}_{\text{AA}}$, $B$ picks $\alpha_x, \beta_x, \gamma_x \in \mathbb{Z}_p$, and $PK_x = \{e(g,g)^{\alpha_x}, g^{\beta_x}, g^{\gamma_x}\}$
  2. Else, let $X = \{x \in \mathcal{I}_{\text{AA}}\}$. $B$ chooses $\tilde{\alpha}_x, \tilde{\beta}_x, \tilde{\gamma}_x \in \mathbb{Z}_p$ and sets implicitly $\alpha_x = \tilde{\alpha}_x + \sum_{i \neq x} b_x \alpha^{x+1} M_{i,x} \cdot \beta_x = \tilde{\beta}_x + \sum_{i \neq x} b_x \beta^{x+1} M_{i,x} \cdot \gamma_x = \tilde{\gamma}_x + \sum_{i \neq x} b_x \gamma^{x+1} M_{i,x}$.

Using the proper terms in q-DBDHE2 tuple $D$, it outputs

$$PK_x = \{e(g,g)^{\alpha_x}, g^{\beta_x}, g^{\gamma_x}\}$$

$$= \{e(g,g)^{\alpha_x} \prod_{x \in X} e(g^{\alpha_x},g^{(\alpha_x)})^{M_{i,x}}, g^{\beta_x} \prod_{x \neq i} (g^{\beta^{x+1}})^{M_{i,x}}, g^{\gamma_x} \prod_{x \neq i} (g^{\gamma^{x+1}})^{M_{i,x}}\} \quad (5.3)$$

- User’s PK and SK: for a user GID, $B$ picks $u_{GID} \in \mathbb{Z}_p$. If $S_{\text{AA}} \cup S_{\text{AA}}$ doesn’t satisfy the access policy, it returns $PK_{GID} = \{g^{\omega}\}$. $SK_{GID} = \{u_{GID}^{-1}\}$ to user GID. Else, it only returns $PK_{GID} = \{g^{\omega}\}$.
- $H_{\text{AA}}$-oracle queries: for a GID that has been queried, $B$ returns the same answer. Else,
  1. If GID doesn’t possess any attributes of the policy, i.e. $\{\rho(x) \notin S_{\text{AA}}\}_{x \in [n]}$, $B$ picks $\tilde{h}_x \in \mathbb{Z}_p$ and returns

$$H_{\text{AA}}(GID) = (g^h_g g^{\omega_1} \cdots g^{\omega_n})^{\omega} = (g^h_g \prod_{e=2}^{n} g^{\omega_e})^{\omega} \quad (5.4)$$
2) Else if, for some row \( x \in [n] \), it’s true \( \rho(x) \in S_{\text{gad}} \) and \( S_{\text{gad}} \cup S_{\text{cud}} \) doesn’t meet the policy, a vector \( d = (1, d_2, \ldots, d_l) \in Z^l_p \) that satisfies the condition \( \{ M'_x \cdot d = 0 \}_{\rho(x) \in \text{gad} \cup \text{cud}} \) can be found. According to lemma 1, the set of corrupted rows in matrix \( M' \) spans the subspace of dimension \( c \). Hence, the vector \( d \) is orthogonal to any of the vectors \((0, \ldots, 0, 0, \ldots, 0) \in Z^l_p \) and \( \{ d_j = 0 \}_{l+1 \leq j \leq d} \). \( B \) picks \( \tilde{h}_1 \leftarrow Z^l_p \) and returns

\[
H_1(GID) = (g_{\tilde{h}_1} g^{ad_1} g^{d_2} \cdots g^{d_l})_{\text{gad}} = (g_{\tilde{h}_1} \prod_{t=2}^l g^{d+t-1})_{\text{gad}} \tag{5.5}
\]

3) Else, \( S_{\text{gad}} \) satisfies the policy and \( E \) picks \( \tilde{h}_1 \leftarrow Z^l_p \) and returns \( H_1(GID) = g_{\tilde{h}_1} \)

- \( H_2 \)-oracle queries: for an attribute \( u \) of GID that has been queried, \( B \) returns the same answer.

Else,

1) If \( AA_u \) doesn’t manage any row of \( M' \), i.e. \( AA_u \notin \delta(n) \), or \( AA_i \) is corrupted, then \( B \) returns a random element of \( G_1 \)

2) Else, \( AA_i \in \delta(n) \), let \( X' \) be the set of rows owning to \( AA \) but not mapping to attribute \( u \), i.e.

\( X' = \{ x \in [n] \mid \delta(x) = AA \land \rho(x) \neq u \} \). \( B \) picks \( \tilde{h}_2 \leftarrow Z^l_p \) and returns

\[
H_2(u) = g_{\tilde{h}_2} \prod_{x \in X'} (g^{b_{xu} M_{xu}} g^{b_{xu} M_{xu}} \cdots g^{b_{xu} M_{xu}})_{M_{xu}} = g_{\tilde{h}_2} \prod_{x \in X'} (g^{b_{xu} M_{xu}})_{M_{xu}} \tag{5.6}
\]

- \( F \)-oracle queries: For a GID that has been queried, \( B \) returns the same answer. Else, \( B \) picks \( \tilde{f} \leftarrow Z^l_p \) and returns \( F(GID) = \tilde{f} \)

- User’s attribute key: for each \( u \in S_i \) of queried \( \{(GID_i, S_i)\}_{i=1}^m \), \( B \) computes the attribute secret key as follows:

1) \( T(u) = AA_i \notin \delta(n) \): the authority of the attribute \( u \) is not present in the challenge policy. \( PK_i, H_1 \) and \( H_2 \) are calculated by the first, first and second methods respectively. \( B \) picks \( t \leftarrow Z^l_p \) and returns

\[
SK_{gi, u} = g^{\text{hagd}} H_i(GID)^{\text{hagd}} H_i(u)^{\text{hagd}} g^{\text{gad}} \tag{5.5}
\]

2) \( T(u) = AA_i \in \delta(n) \) and \( S_i \bigcap \rho(n) = \emptyset \): the AA of the attribute \( u \) is present in the challenge policy, but none of the attributes of this user is in it. \( PK_i, H_1 \) and \( H_2 \) are calculated by the second, first and second methods respectively. \( B \) sets implicitly \( t = -\sum_{k \in [l]} a^k u_{\text{gad}} \) and computes

\[
K_{gad, u} = g^{\text{hagd}} H_i(GID)^{\text{hagd}} H_i(u)^{\text{hagd}} g^{\text{hagd}} \tag{5.5}
\]

\[
K_{gad, u} = g^{\text{hagd}} H_i(GID)^{\text{hagd}} H_i(u)^{\text{hagd}} g^{\text{hagd}} \prod_{x \in X} (g^{b_{xu} M_{xu}})_{M_{xu}} \prod_{x \in X} (g^{b_{xu} M_{xu}})_{M_{xu}} \tag{5.7}
\]

Since \( t \) is not properly distributed, \( B \) re-randomizes this key according to lemma 2 and returns the result.

3) \( T(u) = AA_i \in \delta(n) \) and \( S_i \bigcap \rho(n) \neq \emptyset \): the user holds some shares of the challenge policy. \( PK_i, H_1 \) and \( H_2 \) are calculated all by the second methods respectively. \( B \) sets implicitly \( t = -\sum_{k \in [l]} a^k u_{\text{gad}} \) and computes
\[ K_{\text{GID},a} = g^{\mu_{\text{aa}} H_1(\text{GID})} g^a H_2(a) g^{\phi(\text{GID})} = g^{a \mu_{\text{aa}} H_1(\text{GID})} (g^\beta)^{\mu_{\text{aa}}} (g^{\beta})^{\mu_{\text{aa}}} (g^{\beta})^{\mu_{\text{aa}}} \]  
\[ \prod_{x \in \mathcal{X}} \prod_{j=1}^{t_x} (g^{h_{\text{aa},t_x+1}})^{M_{\text{aa},j}} \prod_{x \in \mathcal{X}} \prod_{j=1}^{t_x} (g^{h_{\text{aa},t_x+1}})^{M_{\text{aa},j}} \prod_{x \in \mathcal{X}} \prod_{j=1}^{t_x} (g^{h_{\text{aa},t_x+1}})^{M_{\text{aa},j}} \]

(5.9)

\[ K_{\text{GID},a} = g' = g^{\sum_{x \in \mathcal{X}} \delta_{\text{aa}} g_{\text{aa}}^{M_{\text{aa},j}}} = \prod_{x \in \mathcal{X}} (g^{a \delta_{\text{aa}}})^{M_{\text{aa},j}} \]  

(5.10)

For a row \( x \in \mathcal{X} \), Else, it picks \( t \in \mathbb{Z}_p \), and computes:

\[ C_{2,s} = H_2((\rho(x')))^t \]

(5.11)

\[ C_{3,s} = g^{r_{\text{aa},t_x}} \]

(5.12)

\[ C_{4,s} = C_{3,s} \]

The equation above can be calculated by the proper terms in q-DPBDHE2 tuple D, but the ciphertext need to re-randomize according to \textbf{lemma 2} since the vectors \( v \) and \( w \) are not properly distributed.

**Guess:** If \( A \) guesses the bit \( b \) correctly, \( B \) outputs \( T = e(g,g)^{\phi_{\text{aa}}} \). Else, it outputs \( T = R \).

When \( T = e(g,g)^{\phi_{\text{aa}}} \), the challenge ciphertext simulates the real ciphertext perfectly and the advantage that \( A \) wins the game is \( \varepsilon \). When \( T = R \), \( A \) knows nothing about the ciphertext and the advantage that he wins the game is 0. Hence, \( B \) can solve the q-DPBDHE2 problem with non-negligible advantage \( \varepsilon / 2 \), which conflicts to the q-DPBDHE2 assumption. So our UR-OD-MA-CPABE construction is statically secure.

\[ 5.2. \text{Forward Security} \]
If the attribute set $S_{gid}$ doesn’t satisfy the access policy $\lambda$, no constants $\{c_i \in Z_q\}_{i \in \lambda}$ can be found such that $\sum_{i \in \lambda} c_i \lambda_i = \nu$ and user GID cannot recover the secret $\nu$ to decrypt the ciphertext. In the user revocation process, we assume that CS will not collude with the revoked users. For a revoked user $GID_i$, since CS has re-encrypt the ciphertext according to the RL and inserts the revoked GIDs into the new ciphertext, he cannot get $D = e(g_1, g)^{p(GID)}$ from his update key and cannot get the plaintext.

5.3. Anti-Collusion Attack

Our construction can prevent the collusion attack between unauthorized (including revoked ) users. According to the construction in [5], each attribute key of the user is generated according to its GID, and different users cannot collide with the attribute key. During the user revocation process, CS embeds the user’s GID into the re-encrypted ciphertext. Although the attribute set satisfies the access policy, the updated ciphertext cannot be decrypted. And multiple revoked users cannot have a greater advantage by colluding. Hence our UR-OD-MA-CPABE can prevent collusion attack.

6. Performance Analysis

In this section, our scheme and related constructions in [4][5][8] are compared in terms of functional characteristics, storage and computation overhead.

6.1. Feature Comparison

As shown in Table 1, although the security of our solution is lower than others, the implemented functions have great advantages in supporting large universe, user revocation and outsourcing decryption. In our scheme, CA only performs user identity authentication, distributes user secret keys and update keys and it doesn’t possess any information, so it won’t be a system bottlenecks. AAs does not interact with each other. So the whole system is still distributed. The schemes in [4][8] built on the composite order group achieve higher security, but the computational efficiency on the composite order group is lower. The functions of the scheme on the prime order group in [5] is not comprehensive enough.

| Scheme            | Large universe | CA | Revocation | Outsourcing decryption | Order of group | Security   |
|-------------------|----------------|----|------------|------------------------|----------------|-----------|
| Lin et al[4]      | x              | x  |            | \( \checkmark \)      | x              | Composite  | Adaptive  |
| Rouselakis et al[5]| x              | \( \checkmark \) | \( \checkmark \) | x                      | Prime          | Selective  |
| Yang et al[8]     | \( \checkmark \) | x  |            | x                      | Composite      | Adaptive   |          |
| ours              | \( \checkmark \) | \( \checkmark \) | \( \checkmark \) | \( \checkmark \)      | Prime          | Static     |

6.2. Computation Overhead

The computation overhead of our scheme and related constructions is listed in table 2 and the meanings of related symbols are shown in Table 3. The multiplication and hash functions are ignored in the table since they take little time contrast to exponential and bilinear operations. Since the number of AAs in the system is much smaller than the total number of attributes, our solution is much more efficient in the system setup phase than the other two solutions. In order to achieve outsourced decryption and ciphertext update, the calculation cost of the key generation and encryption phase is slightly higher, but the user’s decryption only needs one exponential operation, which makes it more feasible for users to access data on some mobile devices.

| Scheme            | AA setup | KeyGen | Encryption | User decryption |
|-------------------|----------|--------|------------|-----------------|
| Lin et al[4]      | \(2|\!E + P\) | \(2|\!E\) | \((5n + 1)|\!E\) | \(|\!E + 2|\!P\) |
| Rouselakis et al[5]| \(2|\!E + P\) | \(2|\!E\) | \((5n + 3r + 1)|\!E\) | \((|\!E + r| + 2|\!E + r|)|\!P\) |
| Yang et al[8]     | \((d + 2|\!E|) + P\) | \(2|\!E\) | \((5n + 1)|\!E\) | \((|\!E + 3|)|\!P\) |
| ours              | \(3dE + P\) | \((5n + 1)|\!E\) | \((7n + 1)|\!E\) | \(|\!E\) |
6.3. Storage Overhead
As shown table 4, the public key in our scheme is much smaller than others. The ciphertext in each scheme is linearly associated with the number of rows in the access matrix. Since CS has strong storage and computing power, the length of ciphertext doesn’t matters. A user can outsource the attribute secret key and store only one user secret key locally. Therefore, our solution has great advantages in the cloud environment.

| Scheme       | Attribute PK | User secret key | Ciphertext |
|--------------|--------------|-----------------|------------|
| Lin et al[4] | $2|U|+d$       | $\lfloor s \rfloor + 1$ | $5n + 2r + 1$ |
| Rouselakis et al[5] | $2|U|+2$   | $\lfloor s \rfloor + 1$ | $3n + 2r + 1$ |
| Yang et al[8] | $2|U|+d$       | $\lfloor s \rfloor$ | $4n + 1$ |
| Ours         | $3d+1$       | 1               | $5n + 2r + 1$ |

7. Conclusion
A multi-authority ABE scheme for sharing data securely in cloud storage environment was proposed. It supports large universe, user revocation and outsourcing decryption, and has great practicality. CS can re-encrypt ciphertext to achieve user revocation and perform partial decryption to reduce the computation overhead of the client. And the construction is proved to be statically secure in random oracle.

8. References
[1] Sahai A and Waters B. Fuzzy Identity-Based Encryption[C]. International Conference on Theory & Applications of Cryptographic Techniques. 2005.
[2] Waters B. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization [J]. 2011.
[3] Lewko A, Waters B. Decentralizing Attribute-Based Encryption[C]. Advances in Cryptology-eurocrypt -international Conference on the Theory & Applications of Cryptographic Techniques. 2011.
[4] Cui H, Deng R H. Revocable and Decentralized Attribute-Based Encryption [J]. Computer Journal, 2016, 59(8):bxw007.
[5] Horvá th, Má té . Attribute-based encryption optimized for cloud computing. [J]. Infocommunications Journal, 2015, 7(2):1-9.
[6] Lin H, Cao Z, Liang X, et al. Secure threshold multi authority attribute based encryption without a central authority [J]. Information Sciences, 2010, 180(13):2618-2632.
[7] Rouselakis Y, Waters B. Efficient statically-secure large-universe multi-authority attribute-based encryption. [C] International Conference on Financial Cryptography & Data Security. Springer, Berlin, Heidelberg, 2015
[8] Yang Y , Chen X , Chen H , et al. Improving Privacy and Security in Decentralizing Multi-Authority Attribute-Based Encryption in Cloud Computing[J]. IEEE Access, 2018:1-1.
[9] Rahulamathavan Y. User Collusion Avoidance Scheme for Privacy-Preserving Decentralized Key-
Policy Attribute-Based Encryption -- Full Version [J]. IEEE Transactions on Computers, 2016, 65(9):2939-2946.

[10] Ruj S, Nayak A, Stojmenovic I. DACC: Distributed Access Control in Clouds[C]. 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11. IEEE Computer Society, 2011.

[11] Yang K, Jia X. Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage [J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(7):1735-1744.

[12] Zhang K, Ma J, Liu J, et al. Adaptively secure multi-authority attribute-based encryption with verifiable outsourced decryption [J]. Science China Information Sciences, 2016, 59(9):99105.

[13] BEIMEL A. Secure Schemes for Secret Sharing and Key Distribution [D]. Haifa,Israel,Israel Institute of Technology, 1996.