Hindawi
Security and Communication Networks
Volume 2019, Article ID 8093578, 11 pages
https://doi.org/10.1155/2019/8093578

*Research Article*

# Efficient Ciphertext-Policy Attribute-Based Online/Offline Encryption with User Revocation

**Haiying Ma**,[1] **Zhanjun Wang**,[2] **and Zhijin Guan**[1]

[1]*School of Computer Science and Technology, Nantong University, Nantong 226019, China*
[2]*School of Science, Nantong University, Nantong 226019, China*

Correspondence should be addressed to Zhanjun Wang; wzj8855@ntu.edu.cn

Received 9 September 2018; Accepted 5 January 2019; Published 14 February 2019

Academic Editor: Petros Nicopolitidis



Attribute-Based Encryption (ABE) must provide an efficient revocation mechanism since a user's private key can be compromised or expired over time. The existing revocable ABE schemes have the drawbacks of heavy computational costs on key updates and encryption operations, which make the entities for performing these operations a possible bottleneck in practice applications. In this paper, we propose an efficient Ciphertext-Policy Attribute-Based Online/Offline Encryption with user Revocation (R-CP-ABOOE). We integrate the subset difference method with ciphertext-policy ABE to significantly improve key-update efficiency on the side of the trusted party from $O(r \log(N/r))$ to $O(r)$, where $N$ is the number of users and $r$ is the number of revoked users. To reduce the encryption burden for mobile devices, we use the online/offline technology to shift the majority of encryption work to the offline phase, and then mobile devices only need to execute a few simple computations to create a ciphertext. In addition, we exploit a novel trick to prove its selective security under the $q$-type assumption. Performance analysis shows that our scheme greatly improves the key-update efficiency for the trusted party and the encryption efficiency for mobile devices.

## 1. Introduction

Attribute-based encryption (ABE) is a promising alternative of encryption for achieving fine-grained access control of encrypted data. The notion of ABE is first proposed by Sahai and Waters [1], and then Goyal et al. [2] formalize two supplementary forms of ABE: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In CP-ABE [3, 4], each user possesses a private key that corresponds to his attribute set. A ciphertext is embedded into an access policy over the possible attributes. The ciphertext can be successfully decrypted by the users whose attributes satisfy the policy. Alternatively, in KP-ABE the roles are swapped: a ciphertext is associated with an access policy and the private key is related to a set of attributes.

Any ABE system must provide an efficient method to revoke users since a user's private key can be compromised or expired over time. As a practical solution to the problem for ABE, Boldyreva et al. [5] used the complete subtree method [6] to revoke users. In their scheme, a user can encrypt with the set of attributes and the current time attribute, e.g., "TIME: 2018. WEEK 16," and the key generation center (KGC), who owns the latest revocation list, periodically broadcasts the key update component at each time slot such that all nonrevoked users can reconstruct their decryption key and utilize it decrypt ciphertexts. Subsequently, Attrapadung and Imai [7] proposed a revocable KP-ABE by using a similar method. However, the key-update work can be a system bottleneck since KGC requires broadcasting key-update component to all nonrevoked users at all time slots. Although their solutions [6, 7] reduce this work from $O(N)$ to $O(r \log(N/r))$, where $N$ is the number of users and $r$ is the number of revoked users and the key-update work still is a bottleneck. Lee et al. [8] proposed an efficiently revocable identity-based encryption using subset difference methods, which has $O(r)$ number of group elements in an update key. We note that their idea cannot be directly used to construct revocable ABE, because a user's identity is not related to decryption in ABE systems.

At the same time, in all revocable ABE schemes, the encryption process must perform a lot of exponentiations, and the encryption cost grows with the complexity of access

policy or number of attributes. If a mobile device performs the encryption task, battery power and encryption time will be a large problem. To significantly reduce the encryption cost for mobile device, a few online/offline ABE schemes [9–12] are proposed, and move the majority of encryption computations into an offline phase. Mobile device only needs to execute a few simple calculations to create a ciphertext. However, the existing online/offline ABE schemes cannot revoke users. Therefore, our goal is to integrate the subset difference method [6, 8] with ABE to significantly decrease the key update work, and use the online/offline technique to greatly improve the encryption efficiency for mobile devices.

In this work, we propose an efficient ciphertext-policy attribute-based online/offline encryption with user revocation. In particular, our contributions have three aspects as follows:

(1) We present a novel technique that may integrate CP-ABE with the Subset Difference (SD) methodology to significantly improve the key-update efficiency for KGC. In SD scheme, an assigned key for each subset is dependent on other keys, so we may categorize all subsets into different groups. For each group $GT$ in SD, we assign a random polynomial $f_{GT}(x) = a_{GT}x + \alpha$ to it, where $a_{GT}$ is a random value that is uniquely assigned to the group $GT$ and $\alpha$ is the master secret key. Using this linear polynomial, we split the master secret key $\alpha$ into two shares $f_{GT}(1)$ and $f_{GT}(T)$, where $T$ is an update time and is a greater than 1 positive integer. The share $f_{GT}(1)$ is used to build a private key which is related to user's attribute set in CP-ABE as usual, and the other share $f_{GT}(T)$ is used to build an update key which is related to the update time $T$. Then KGC periodically broadcasts the update keys, and any nonrevoked user can reconstruct his decryption key. Since all revoked users possess the same share $f_{GT}(1)$, they cannot collude to reconstruct a decryption key. Therefore, our scheme may provide the security against collusion attacks.

(2) To greatly reduce the encryption cost for mobile devices, we employ the online/offline technique of [8] to split encryption process into the offline phase and the online phase. The offline encryption first performs a majority of encryption task before the plaintext and its access policy are known. Once the specifications are given, the online encryption only executes a few simple calculations to produce a ciphertext by using the offline-ciphertexts pool that is similar to the reference [8]. Especially, the offline work can be executed by a high-performance computer, and then lightweight devices can quickly run the online encryption on the move without greatly draining the battery.

(3) We present a new method to prove the security of our scheme. Based on the CP-ABE's access policy $\boldsymbol{A}^*$ and a challenge update time $T^*$, we construct a new challenge access policy $\boldsymbol{A}^{**} = \boldsymbol{A}^* \wedge T^*$, *i. e.*, we may obtain $\boldsymbol{A}^{**}$ by appending a well-designed vector to $\boldsymbol{A}^*$. Next, we can create the public parameters needed in the proof. Moreover, unlike the complete subtree method, the intersection of the covering collection of nonrevoked users and the private key of the revoked users is not empty. It leads to impossibility to construct the update key on the time $T^*$. We solve this problem by reasonably assigning the users' location in the binary tree. Our scheme is proved to be selectively secure under the *q-type* assumption. The above method may be of independent interest in the security proof of the revocable CP-ABE scheme. Compared with the related works, our scheme can significantly improve the key-update efficiency for the trusted party from $O(r \log(N/r))$ to $O(r)$ and the encryption efficiency for mobile devices, where $N$ is the number of users and $r$ is the number of revoked users, and greatly decrease the number of group elements in the update key.

ABE is a useful cryptographic technology to protect private data and achieve fine-grained access control simultaneously [12, 13], many variants of ABE were proposed to realize promising properties, *e. g.*, efficient large universe ABE [4], ABE with verifiable outsourced decryption [14–16], traceable-then-revocable ABE [17, 18], CP-ABE against key-delegation abuse [19], the fully-secure ABE [11], ABE resilience against continuous auxiliary-inputs leakage [20, 21].

To reduce the encryption cost of ABE, a few works [9–12] presented the online/offline ABE by the online/offline cryptography [22–25], but their schemes cannot revoke users. To satisfy the practical requirement that users' access rights can be revoked dynamically, many revocable ABE schemes have been proposed [10–20]. The core idea of the works [5, 7, 26–31] utilizes the complete subtree scheme of NNL [8] to revoke user's private key. Lee et al. [8] proposed a revocable identity-based encryption to reduce the size of the update key. Sahai et al. [29] constructed a few ABE schemes of revocable storage by providing a ciphertext delegation means without any interaction with data owners. Datta et al. [32] uses the subset difference to directly revoke users in KP-ABE systems. Nevertheless, all the existing revocable ABE schemes have the drawbacks of heavy computational costs on key updates and encryption operations.

Therefore, it will be indispensable to reduce the heavy computational overhead on the key update work and the encryption task. Different from prior works, we integrate the SD method with the online/offline technique in the CP-ABE system, which not only may efficiently revoke users, but also can significantly improve the key-update efficiency and encryption efficiency.

We review some preliminaries in Section 2. The definition and security model of our scheme is defined in Section 3. We propose an efficient ciphertext-policy attribute-based online/offline encryption with user revocation and prove its selective security in Section 4. Performance analysis is given in Section 5. Finally, we conclude this work in Section 6.

## 2. Preliminaries

In this section, we first elaborate the definitions of bilinear group and the complexity assumption for our R-CP-ABOOE scheme. Then we state a brief review of access structures and linear secret-sharing schemes (LSSS). Finally, we introduce the notions of full binary tree and the subset difference method.

*2.1. Bilinear Group and the Complexity Assumptions.* We give some notations. For $n \in \mathbb{N}$, we define $[n] = \{1, 2, \ldots, n\}$, and for $n_1, n_2, \ldots, n_k \in \mathbb{N}$, $[n_1, n_2, \ldots, n_k] = [n_1] \times [n_2] \times \cdots \times [n_k]$. For $l, n \in \mathbb{N}$, by $Z_p^{l \times n}$ we denote a set of matrices of size $l \times n$ with elements in $Z_p$. A row vector is denoted as $\langle y_1, y_2, \ldots, y_n \rangle$, while a column vector is denoted as $\langle y_1, y_2, \ldots, y_n \rangle^{\mathrm{T}}$, where T denotes the transpose of the vector.

*Definition 1* (bilinear group). Let $G$, $G_{\mathrm{T}}$ be multiplicative cyclic groups of prime order $p$, and we define a symmetric bilinear map $e$: $G \times G \longrightarrow G_{\mathrm{T}}$ with the following properties: (1) bilinearity: for all $g_1, g_2 \in G$, and $c, d \in Z_p$, we have $e(g_1^c, g_2^d) = e(g_1, g_2)^{cd}$; (2) nondegeneracy: $\exists u \in G$, $e(u, u) \neq 1$; (3) computability: the bilinear map $e$ and the group operations in $G$ and $G_{\mathrm{T}}$ are efficiently computable. Then the tuple $(p, G, G_{\mathrm{T}}, e)$ is called a bilinear group.

*Definition 2* ($q$-type assumption [4]). Given a security parameter $\kappa \in \mathbb{N}$, an integer $q$, a group generator $\mathcal{G}(\kappa)$ that outputs the bilinear group description $(p, G, G_{\mathrm{T}}, e)$, chooses a random element $g \in G$ and $q + 2$ random values $a, s, b_1, b_2, \ldots, b_q \in Z_p$ and computes the following terms $E$:

$$
\begin{gathered}
g, g^s; \\
g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{a^i / b_j^2} \quad \forall (i, j) \in [q, q]; \\
g^{a^i / b_j} \quad \forall (i, j) \in [2q, q] \text{ and } i \neq q + 1; \\
g^{a^i b_j / b_{j'}^2} \quad \forall \left(i, j', j\right) \in [2q, q, q] \text{ and } j \neq j'; \\
g^{sa^i b_j / b_{j'}}, g^{sa^i b_j / b_{j'}^2} \quad \forall \left(i, j', j\right) \in [q, q, q] \text{ and } j \neq j'.
\end{gathered} \tag{1}
$$

If any probabilistic polynomial-time (PPT) adversary obtains the bilinear group description $(p, G, G_{\mathrm{T}}, e)$ and the above terms $E$, it cannot distinguish the element $F = e(g, g)^{sa^{q+1}} \in G_{\mathrm{T}}$ from a random element $R \in G_{\mathrm{T}}$ with a nonnegligible advantage. We say that the *q-type* assumption holds in the bilinear group $(p, G, G_{\mathrm{T}}, e)$.

*2.2. Access Structures and Linear Secret Sharing Scheme (LSSS)*

*Definition 3* (access structures [4]). Let $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k\}$ be a set of attributes. A collection $\mathbb{A} \subseteq 2^{\mathcal{I}}$ is monotone if $\forall C, D$ and $C \in \mathbb{A}, C \subseteq D$ then $D \in \mathbb{A}$. An access structure is a collection $\mathbb{A}$ of nonempty subsets of $\mathcal{I}$, *i.e.*, $\mathbb{A} \subseteq 2^{\mathcal{I}} \setminus \{\varnothing\}$. The authorized sets are defined as the sets in $\mathbb{A}$, and the unauthorized sets are the sets not in $\mathbb{A}$.

*Definition 4* (LSSS [4]). Let $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k\}$, a function $\rho$: $\{1, 2, \ldots, l\} \longrightarrow \mathcal{I}$. A secret-sharing scheme $\Pi$ over a set of attributes $\mathcal{I}$ is called linear over $Z_p$ if (1) the shares for each attribute form a vector over $Z_p$; (2) there is a matrix $\boldsymbol{A}$ called the share-generation matrix for $\Pi$. The matrix $\boldsymbol{A}$ has $l$ rows and $n$ columns. For each row $i = 1, 2, \ldots, l$, the $i^{\text{th}}$ row of $\boldsymbol{A}$ is labeled by an attribute $\rho(i)$. Let the column vector $\boldsymbol{v} = \langle s, r_2, \ldots, r_n \rangle^{\mathrm{T}}$, where $s \in Z_p$ is the shared secret and $r_2, \ldots, r_n \in Z_p$ are randomly picked, then $\boldsymbol{A} \bullet \boldsymbol{v}$ is the vector of $l$ shares of the secret $s$ according to $\Pi$. The share $(\boldsymbol{Av})_i$ belongs to the attribute $\rho(i)$. Reconstruction property. Suppose the scheme $\Pi$ is an LSSS for the access structure $\mathbb{A}$. Let $\omega \in \mathbb{A}$ be any authorized set, and $I \subseteq \{1, 2, \ldots, l\}$ is defined as $I = \{i : \rho(i) \in \omega\}$. If $\{s_i\}$ are valid shares of any secret $s$, there are constants $\{c_i \in Z_p\}_{i \in I}$ that satisfy $\sum_{i \in I} c_i s_i = s$.

*2.3. Full Binary Tree.* We follow the terminologies on the full binary tree in [8]. Let $\mathcal{T}$ be a full binary tree with $N$ leaves. Each user $ID$ is assigned to a unique leaf node $id$. In the tree, it has $2N$-1 nodes, for $\eta = 1$ to $2N$-1, let $v_\eta$ denote a node, and each edge is labelled as a bit 1 or 0, at the right branch corresponds to bit 1 and the left branch corresponds to bit 0. The identifier $L_\eta$ of a node $v_\eta$ is defined as the unique bitstring which is the bitstring of all edges in the path from the root to the node $v_\eta$. For each node $v_\eta$, its depth $d_\eta$ is the size of the path from the root to $v_\eta$.

Let $S_\eta$ denote the set of leaves in the subtree that is rooted at the node $v_\eta$. For any two nodes $v_\eta$, $v_\theta$ so that $v_\eta$ is an ancestor node of $v_\theta$, $S_{\eta,\theta}$ is defined as the set of all leaves that are descendants of $v_\eta$ but not $v_\theta$, *i.e.*, $S_{\eta,\theta} = S_\eta - S_\theta$.

Let $GT$ identify a group, and a group's tag $GT$ is described as the collection of subset $S_{\eta,\theta}$ such that $v_\eta$ is the same node and other different nodes $v_\theta$ have the same depth, and let $GT = L_\eta \parallel d_\theta$. In our scheme, we randomly choose $a_{GT} \in Z_p$ and specify a polynomial $f_{GT}(x) = a_{GT} x + \alpha$ once for one group $GT$ to revoke users.

Let $\mathcal{R}$ be the list of all revoked users, $ST(\mathcal{R})$ be the Steiner Tree induced by the root node and the set $\mathcal{R}$, that is, the minimal subtree of the tree $\mathcal{T}$ connects the root node and all the leaves in $\mathcal{R}$.

*2.4. Subset Difference Method.* As a general revocation methodology, Subset-Cover framework [6] includes both the complete subtree method and the subset difference (SD) method, and SD was presented as an improvement on the complete subtree scheme. The definition of SD is given as follows.

*Definition 5* (subset difference). The SD scheme for the set $\mathcal{N} = \{1, 2, \ldots, N\}$ of all users consists of four algorithms: Setup, Assign, Cover, Match, which are described as following:

**SD.Setup**$(\mathcal{N}) \longrightarrow (\mathcal{T}, \mathcal{S})$: The setup algorithm inputs all users $\mathcal{N}$ and sets a full binary tree $\mathcal{T}$ with at least $N$ leaves, where $N$ is the total number of users. It assigns a unique leaf of $\mathcal{T}$ once to a user. For any two nodes $v_\eta, v_\theta \in \mathcal{T}$ such that $v_\eta$ is an ancestor node of $v_\theta$ and it gets the collection $\mathcal{S}$ of subset $\{S_{\eta,\theta}\}$. It outputs the tree $\mathcal{T}$ and the collection $\mathcal{S}$.

**SD.Assign**$(\mathcal{T}, ID) \longrightarrow PV_{ID}$: Given the full binary tree $\mathcal{T}$ and a user $ID \in \mathcal{N}$, the assignment algorithm assigns a leaf $id$ of $\mathcal{T}$ to the user $ID$. Let $(v_0, v_{k_1}, \ldots, v_{k_n} = id)$ denote the path from the root $v_0$ to the leaf $id$ and a private set $PV_{ID}$ be an empty one. For all $v_\eta, v_\theta \in \{v_0, v_{k_1}, \ldots, v_{k_n}\}$ such that $v_\eta$ is an ancestor of $v_\theta$ and it adds all subset $S_{\eta,\theta}$ to $PV_{ID}$. Then it outputs the user's private collection $PV_{ID} = \{S_{\eta,\theta}\}$.

**SD.Cover**$(\mathcal{T}, \mathcal{R}) \longrightarrow CV_{\mathcal{R}}$: Given the full binary tree $\mathcal{T}$ and the set $\mathcal{R}$ of revoked users, the covering algorithm

gets the Steiner Tree $ST(\mathcal{R})$ and sets a subtree $TR = ST(\mathcal{R})$ and then constructs a covering collection $CV_{\mathcal{R}}$ by iteratively removing nodes from the tree $TR$ until $TR$ is made up of only a single node as follows:

(1) It derives two leaves $v_i$ and $v_j$ from the tree $TR$ and then finds their least-common ancestor $v$ which does not contain any other leaf in this tree $TR$. Let $v_l$ and $v_k$ be two child of $v$ where $v_l$ is a father node of $v_i$ and $v_k$ is a father node of $v_j$. If $TR$ has only one leaf, it sets $v_i = v_j$ to be a leaf node and sets $v = v_l = v_k$ to be the root of $v$.

(2) If $v_k \neq v_j$, it adds the subset $S_{k,j}$ to $CV_{\mathcal{R}}$; similarly, if $v_l \neq v_i$, it adds the subset $S_{l,i}$ to $CV_{\mathcal{R}}$.

(3) It deletes all descendants of $v$ from $TR$ and lets $v$ be a leaf.

It outputs the covering collection $CV_{\mathcal{R}} = \{S_{i,j}\}$.

**SD.Match**$(PV_{ID}, CV_{\mathcal{R}}) \longrightarrow (S_{\eta,\theta}, S_{i,j})/\perp$: Given a private collection $PV_{ID} = \{S_{\eta,\theta}\}$ and a covering collection $CV_{\mathcal{R}} = \{S_{i,j}\}$, this matching algorithm searches two subsets $S_{\eta,\theta} \in PV_{ID}$ and $S_{i,j} \in CV_{\mathcal{R}}$ so that $\eta = i, \theta \neq j$, and the nodes $v_\theta$ and $v_j$ have the same depth, i.e., $d_\theta = d_j$. If it finds two subsets, then it outputs $(S_{\eta,\theta}, S_{i,j})$. Otherwise, it fails.

*Remark*. In the SD scheme, the revocation list $\mathcal{R}$ cannot be empty. Let $r$ be the size of $\mathcal{R}$, *i.e.*, $r \geq 1$. To address the case $r = 0$, we may add a dummy user that is revoked so that $|\mathcal{R}'| = r' = r + 1$.

## 3. Definition and Security Model of R-CP-ABOOE

The R-CP-ABOOE scheme consists of eight algorithms: Setup, KeyGen, KeyUpdate, DecKey, $\text{Enc}^{\text{off}}$, $\text{Enc}^{\text{on}}$, Dec, and Rev, which are defined as follows:

**Setup**$(\kappa, \mathcal{U}, \mathcal{N}) \longrightarrow (pk, msk, \Lambda, \mathcal{R})$: This setup algorithm inputs a security parameter $\kappa$, the universe of attributes $\mathcal{U}$, and all users $\mathcal{N}$ and outputs the system public key $pk$, the master secret key $msk$, the state $\Lambda$ and the revocation list $\mathcal{R}$.

**KeyGen**$(msk, (ID, \omega), \Lambda, pk) \longrightarrow sk_{(ID,\omega)}$: This key-generation algorithm inputs the master secret key $msk$, a user identity-attribute pair $(ID, \omega)$, the state $\Lambda$, and public key $pk$ and outputs the user's private key $sk_{(ID,\omega)}$.

**KeyUpdate**$(T, \mathcal{R}, msk, \Lambda, pk) \longrightarrow (\Lambda, uk_{T,\mathcal{R}})$: This key-update algorithm inputs a time $T$, the revocation list $\mathcal{R}$, the master secret key $msk$, the current state $\Lambda$, and the public key $pk$ and outputs the updated state $\Lambda$ and an update key $uk_{T,\mathcal{R}}$.

**DecKey**$(sk_{(ID,\omega)}, uk_{T,\mathcal{R}}, pk) \longrightarrow dk_{(ID,\omega),T} \setminus \perp$: This decryption-key algorithm inputs a user's private key $sk_{(ID,\omega)}$, an update key $uk_{T,\mathcal{R}}$, and the public key $pk$ and outputs a decryption key $dk_{(ID,\omega),T}$ or a failure $\perp$.

$\mathbf{Enc}^{\mathbf{off}}(pk) \longrightarrow (OC_{main}, OC_{att})$: This offline-encryption algorithm can independently create an arbitrary number of main $OC_{main}$ and many attribute modules $OC_{att}$ by using the public key $pk$ and then stores them in a pool of offline-ciphertexts. This algorithm can be performed by high-performance computer.

$\mathbf{Enc}^{\mathbf{on}}((OC_{main}, OC_{att}), m, \mathbb{A}, T) \longrightarrow CT_{\mathbb{A},T}$: This online-encryption algorithm first picks one main module $OC_{main}$ and some attribute modules $OC_{att,\tau}$ from the pool of offline-ciphertexts. Then it inputs a message $m$, an access policy $\mathbb{A}$, and a time $T$ and outputs a ciphertext $CT_{\mathbb{A},T}$. This algorithm is a true encryption process and is performed by the constrained-computation devices.

**Dec**$(CT_{\mathbb{A},T}, dk_{(ID,\omega),T}, pk) \longrightarrow m \setminus \perp$: This decryption algorithm inputs a ciphertext $CT_{\mathbb{A},T}$, a decryption key $dk_{(ID,\omega),T}$, and the public key $pk$ and outputs a message $m$ or a failure notation $\perp$.

**Rev**$(ID, T, \mathcal{R}, \Lambda) \longrightarrow \mathcal{R}'$: This revocation algorithm inputs an identity $ID$, a revocation time $T$, the current revocation list $\mathcal{R}$, and the state $\Lambda$ and outputs the updated revocation list $\mathcal{R}'$.

The correctness of R-CP-ABOOE: For **Setup**$(\kappa, \mathcal{U}, \mathcal{N}) \longrightarrow (pk, msk, \Lambda, \mathcal{R})$, **KeyGen**$(msk, (ID, \omega), \Lambda, pk) \longrightarrow sk_{(ID,\omega)}$, **KeyUpdate**$(T, \mathcal{R}, msk, \Lambda, pk) \longrightarrow (uk_{T,\mathcal{R}}, \Lambda)$, $\mathbf{Enc}^{\mathbf{on}}(\mathbf{Enc}^{\mathbf{off}}(pk), m, \mathbb{A}, T) \longrightarrow CT_{\mathbb{A},T}$, it is satisfied by the two cases:

(1) If $(ID \notin \mathcal{R})$, then **DecKey**$(sk_{(ID,\omega)}, uk_{T,\mathcal{R}}, pk) \longrightarrow dk_{(ID,\omega),T}$. Otherwise **DecKey**$(sk_{(ID,\omega)}, uk_{T,\mathcal{R}}, pk) \longrightarrow \perp$.

(2) If $(\omega \in \mathbb{A}) \wedge (T = T')$, then **Dec**$(CT_{\mathbb{A},T'}, dk_{(ID,\omega),T}, pk) \longrightarrow m$. Otherwise **Dec**$(CT_{\mathbb{A},T'}, dk_{(ID,\omega),T}, pk) \longrightarrow \perp$.

The selective security of R-CP-ABOOE is formally described as the following game between a challenger $\mathcal{S}$ and an adversary $\mathcal{A}$:

**Init**: The adversary $\mathcal{A}$ submits the challenge policy $\mathbb{A}^*$, the challenge time $T^*$, the revocation list $\mathcal{R}^*$ on the challenge time $T^*$ to the challenger $\mathcal{S}$.

**Setup**: $\mathcal{S}$ runs **Setup**$(\kappa, \mathcal{U}, \mathcal{N}) \longrightarrow (pk, msk, \Lambda, \mathcal{R})$ and sends $pk$ to $\mathcal{A}$.

**Phase 1**: $\mathcal{A}$ adaptively requests a polynomial number of the following oracles simulated by $\mathcal{S}$:

(1) The private Key Generation oracle $\boldsymbol{\mathcal{KG}}(\cdot)$. $\mathcal{A}$ submits an identity-attribute pair $(ID, \omega)$ to $\mathcal{S}$, then $\mathcal{S}$ runs **KeyGen**$(msk, (ID, \omega), \Lambda, pk) \longrightarrow sk_{(ID,\omega)}$ and sends $sk_{(ID,\omega)}$ to $\mathcal{A}$.

(2) The Key Update oracle $\boldsymbol{\mathcal{KU}}(\cdot)$. $\mathcal{A}$ submits a time $T$ to $\mathcal{S}$, then $\mathcal{S}$ runs **KeyUpdate**$(T, \mathcal{R}, msk, \Lambda, pk) \longrightarrow (uk_{T,\mathcal{R}}, \Lambda)$ and sends $uk_{T,\mathcal{R}}$ to $\mathcal{A}$.

(3) The Revocation oracle $\boldsymbol{\mathcal{R}}(\cdot)$. $\mathcal{A}$ submits an identity-time pair $(ID, T)$ to $\mathcal{S}$, then $\mathcal{S}$ runs **Rev**$(ID, T, \mathcal{R}, \Lambda) \longrightarrow \mathcal{R}'$ and sends $\mathcal{R}'$ to $\mathcal{A}$.

This phase must be satisfied by the restricted condition as follows: (1) for each query $(ID, \omega)$, where $\omega$ satisfies the challenge policy $\mathbb{A}^*$ and $\boldsymbol{\mathcal{R}}(\cdot)$ must be queried on $(ID, T)$ for any $T \leq T^*$. (2) $\boldsymbol{\mathcal{KU}}(\cdot)$ and $\boldsymbol{\mathcal{R}}(\cdot)$ are queried on the time which cannot be less than the time of all previous queries.

**Challenge**: $\mathcal{A}$ submits two messages $m_0$ and $m_1$ with equal length to $\mathcal{S}$. $\mathcal{S}$ picks a random bit $c \in \{0, 1\}$ and runs $\mathbf{Enc}^{\mathbf{on}}(\mathbf{Enc}^{\mathbf{off}}(pk), T, \mathbb{A}^*, m_c) \longrightarrow CT^*$ and sends $CT^*$ to $\mathcal{A}$.

**Phase 2**: The same as Phase 1.

**Guess**: $\mathcal{A}$ outputs a guess bit $c'$.

Only if $c' = c$, does $\mathcal{A}$ win this game. Therefore, we define the advantage of $\mathcal{A}$ as $\text{Adv}(\mathcal{A}) = |Pr|c' = c| - 1/2|$ in the above game.

*Definition 6.* Our scheme is selectively secure if any PPT adversary can break the above game with negligible advantage.

## 4. Our Scheme and Security Proof

*4.1. Our Intuition.* At a high level we explain how to construct our scheme. We uses the full binary tree with $N$ leaves as in the subset difference (SD) method [8]. Each user $ID$ can be assigned to a leaf node $id$ which is not appointed yet. In SD, all the subsets can be categorized into different groups, and each group $GT$ is assigned a random secret one-degree polynomial $f_{GT}(x)$ such that $f_{GT}(0) = \alpha$, where $\alpha$ is the master secret key. Let $X = e(g,g)^s$. Our scheme uses $X^\alpha = e(g,g)^{\alpha s} \in G_T$ to encrypt a message, where $s$ is a random number.

Given the revoked users' set $\mathcal{R}$ on the time $T$, the KGC utilizes the master secret key to build an update key corresponding to each subset in Cover($\mathcal{R}$). Only the nonrevoked user $ID$ can retrieve a subset $S_{\eta,\theta} \in \text{Cover}(R)$ such that $ID \in S_{\eta,\theta}$, and the user $ID$ can compute $X^{f_{GT}(T)}$. Simultaneously, only the user $ID$ whose attribute set $\omega$ satisfies the access policy associated with ciphertext can compute $X^{f_{GT}(1)}$. Because the function $f_{GT}(x)$ is a first-order polynomial, using these two elements, we can obtain $X^{f_{GT}(0)}$ by interpolation in the exponent. Obviously, the revoked users only can obtain the same element $X^{f_{GT}(1)}$, and then they cannot get two different elements of the one-degree polynomial $f_{GT}(x)$. Therefore, our scheme provides security against collusion attacks.

*4.2. Our Construction.* For $x, i \in Z_p$ and a set of indexes $I \subseteq Z_p$, a Lagrange coefficient $\Delta_{i,I}(x)$ may be defined as

$$\Delta_{i,I}(x) = \prod_{j \in I, j \neq i} \frac{x-j}{i-j}. \tag{2}$$

Our R-CP-ABOOE scheme is described as follows:

**Setup**($\kappa, \mathcal{U}, \mathcal{N}$) $\longrightarrow$ ($pk, msk, \Lambda, \mathcal{R}$): This setup algorithm first runs the group generator $\mathcal{G}(\kappa)$ to get bilinear groups $G$, $G_T$ of prime order $p$, $\mathcal{U} = Z_p$, and randomly picks $g, h, u, v, w \in G$, and $\alpha \in Z_p$. Let a user list $\mathcal{L}_{ID}$ contain a tuple $(ID, id)$ as an empty one, and let a function list $\mathcal{L}_f$ contain a tuple $(GT, f_{GT}(x))$ for a group's tag $GT$ as an empty one. Let $\mathcal{M}$ be the message space and the size of a message $|\mathcal{M}| = 2^{n_\mathcal{M}}$. Let a hash function $H$: $\{0,1\}^* \longrightarrow \{0,1\}^{n_\mathcal{M}}$. It runs SD.Setup($\mathcal{N}$) to get the full binary tree $\mathcal{T}$. Let $\mathcal{C}$ denote the collection of all sets $S_{\eta,\theta}$ of $\mathcal{T}$. For each $S_{\eta,\theta} \in \mathcal{C}$, it makes $GT = L_\eta \parallel d_\theta$ and runs the following: if $(GT, *) \notin \mathcal{L}_f$, then it chooses randomly $a_{GT} \in Z_p$ and specifies a polynomial $f_{GT}(x) = a_{GT}x + \alpha$ once for one group $GT$ and saves $(GT, f_{GT}(x))$ into $\mathcal{L}_f$. Finally, it sets an empty revocation list $\mathcal{R}$, a state $\Lambda = (\mathcal{T}, \mathcal{L}_{ID})$, and outputs a master secret key $msk = (\mathcal{L}_f, \alpha)$ and system public key $pk = (p, g, h, u, v, w, \Phi = e(g,g)^\alpha, H(\cdot))$.

**KeyGen**($msk, (ID, \omega), \Lambda, pk$) $\longrightarrow$ ($sk_{(ID,\omega)}, \Lambda$): This key-generation algorithm inputs the master secret key $msk$, a user identity-attribute pair $(ID, \omega)$ where the set $\omega = (A_1, A_2, \ldots, A_{|\omega|})$, the state $\Lambda = (\mathcal{T}, \mathcal{L}_{ID})$ and public key $pk$. For the user $ID$, it chooses an unassigned leaf node $id$ from $\Lambda$ and stores the tuple $(ID, id)$ into $\mathcal{L}_{ID}$, and runs SD. Assign($\mathcal{T}, ID$) to obtain $PV_{ID} = \{S_{\eta,\theta}\}$. Next, for each $S_{\eta,\theta} \in PV_{ID}$, it sets the group $GT = L_\eta \parallel d_\theta$ and retrieves $(GT, f_{GT}(x))$ from $\mathcal{L}_f$. Second, it chooses random values $r, r_1, r_2, \ldots, r_{|\omega|} \in Z_p$ and computes $K_0 = g^{f_{GT}(1)} w^r$, $K_1 = g^r$. For $\tau = 1$ to $|\omega|$, it computes $K_{2,\tau} = g^{r_\tau}$, $K_{3,\tau} = (u^{A_\tau} h)^{r_\tau} v^{-r}$ and outputs

$$psk_{(ID,\omega),S_{\eta,\theta}} = \left(K_0, K_1, \{K_{2,\tau}, K_{3,\tau}\}_{\tau \in [|\omega|]}\right). \tag{3}$$

Finally, it outputs the updated state $\Lambda$ and a private key

$$sk_{(ID,\omega)} = \left((ID,\omega), PV_{ID}, \left\{psk_{(ID,\tau),S_{\eta,\theta}}\right\}_{S_{\eta,\theta} \in PV_{ID}}\right). \tag{4}$$

**KeyUpdate**($T, \mathcal{R}, msk, \Lambda, pk$) $\longrightarrow$ ($\Lambda, uk_{T,\mathcal{R}}$): This algorithm inputs an update time $T$, the revocation list $\mathcal{R}$, the master secret key $msk$, the state $\Lambda = (\mathcal{T}, \mathcal{L}_{ID})$, and the public key $pk$. First it defines the identities of revoked users on the time $T$ according to $\mathcal{R}$, and uses $\mathcal{L}_{ID}$ to define the revoked index set $RI$. It runs SD.Cover($\mathcal{T}, \mathcal{R}$) to obtain $CV_{RI} = \{S_{\eta,\theta}\}$. Second, for each $S_{\eta,\theta} \in CV_{RI}$, it sets $GT = L_\eta \parallel d_\theta$ and retrieves $(GT, f_{GT}(x))$ from $\mathcal{L}_f$. Then it chooses random values $t, t_1 \in Z_p$ and computes a time-constrained update key

$$\begin{aligned} &tuk_{T,S_{\eta,\theta}} \\ &= \left\{U_0 = g^{f_{GT}(T)} w^t, U_1 = g^t, U_2 = g^{t_1}, U_3 = \left(u^T h\right)^{t_1} v^{-t}\right\}. \end{aligned} \tag{5}$$

Finally, it outputs the updated state $\Lambda$ and an update key

$$uk_{T,\mathcal{R}} = \left(CV_{RI}, \left\{tuk_{T,S_{\eta,\theta}}\right\}_{S_{\eta,\theta} \in CV_{RI}}\right). \tag{6}$$

**DecKey**($sk_{(ID,\omega)}, uk_{T,\mathcal{R}}, pk$) $\longrightarrow$ $dk_{(ID,\omega),T} \setminus \perp$: This decryption-key generation algorithm inputs a user private key $sk_{(ID,\omega)}$, an update key $uk_{T,\mathcal{R}}$ and $pk$. First, if $ID \notin \mathcal{R}$, then it calls $SD.Match(CV_{RI}, PV_{ID})$ to get $(S_{\eta,\theta}, S_{\eta',\theta'})$ such that $S_{\eta,\theta} \in CV_{RI}$, $S_{\eta',\theta'} \in PV_{ID}$, $\eta = \eta' \wedge d_\theta = d_{\theta'} \wedge \theta \neq \theta'$. Otherwise, it exports $\perp$.

Second, it retrieves $tuk_{T,S_{\eta,\theta}}$ from $uk_{T,\mathcal{R}}$ and $psk_{(ID,\omega),S_{\eta',\theta'}}$ from $sk_{(ID,\omega)}$. Since $\eta = \eta' \wedge d_\theta = d_{\theta'}$, $tuk_{T,S_{\eta,\theta}}$, and $psk_{(ID,\omega),S_{\eta',\theta'}}$ share the same $f_{GT}(x)$ for $GT = L_\eta \parallel d_\theta$. It sets $I = \{1, T\}$ and computes two Lagrange coefficients $\Delta_{1,I}(0)$ and $\Delta_{T,I}(0)$. It chooses random values $t', t_1', r', r_1', r_2', \ldots, r_{|\omega|}' \in Z_p$ and computes a decryption key as

$$\begin{aligned} E_0 &= U_0^{\Delta_{T,I}(0)} w^{t'}, \\ E_1 &= U_1^{\Delta_{T,I}(0)} g^{t'}, \\ E_2 &= U_2^{\Delta_{T,I}(0)} g^{t_1'}, \\ E_3 &= U_3^{\Delta_{T,I}(0)} \left(u^T h\right)^{t_1'} v^{-t'}, \\ D_0 &= K_0^{\Delta_{1,I}(0)} w^{r'}, \\ D_1 &= K_1^{\Delta_{1,I}(0)} g^{r'}. \end{aligned} \tag{7}$$

For $\tau = 1$ to $|\omega|$, it computes

$$D_{2,\tau} = K_{2,\tau}^{\Delta_{1,I}(0)} g^{r_\tau'},$$
$$D_{3,\tau} = K_{3,\tau}^{\Delta_{1,I}(0)} \left(u^{A_\tau} h\right)^{r_\tau'} v^{-r'}. \quad (8)$$

Finally, it outputs a decryption-key $dk_{(ID,\omega),T} = ((ID,\omega), D_0, D_1, (D_{2,\tau}, D_{3,\tau})_{\tau\in[|\omega|]}, E_0, E_1, E_2, E_3)$.

**Enc**$^{\mathbf{off}}(pk) \longrightarrow (OC_{main}, OC_{att})$: This offline-encryption algorithm can construct an arbitrary number of main modules $OC_{main}$ and attribute modules $OC_{att}$ by using $pk$. It first choose randomly $s \in Z_p$ and computes $key = e(g,g)^{\alpha s}$ and $C_0 = g^s$. It sets the main module $OC_{main} = (s, key, C_0)$. Then it picks randomly $x_1, x_2, x_3 \in Z_p$ and computes $C_1 = w^{x_1} v^{x_3}$, $C_2 = (u^{x_2} h)^{-x_3}$, and $C_3 = g^{x_3}$ to obtain an attribute module $OC_{att} = (x_1, x_2, x_3, C_1, C_2, C_3)$. Using the above process, it can obtain a lot of main modules $OC_{main}$ and attribute modules $OC_{att}$.

**Enc**$^{\mathbf{on}}(OC_{main}, \{OC_{att,\tau}\}_{\tau\in[l+1]}, m, \mathbb{A}(\boldsymbol{A},\rho), T) \longrightarrow CT_{\mathbb{A},T}$: Suppose that $\boldsymbol{A}$ is an $l\times n$ access matrix, this online-encryption algorithm first picks one main module $OC_{main} = (s, key, C_0)$ and $l+1$ attribute modules $OC_{att} = (x_1, x_2, x_3, C_1, C_2, C_3)$ available from the pool. For $\tau = 1, 2, \ldots, l+1$, it sets the attribute module $OC_{att,\tau} = (x_{1,\tau}, x_{2,\tau}, x_{3,\tau}, C_{1,\tau}, C_{2,\tau}, C_{3,\tau})$. It randomly chooses $y_2, \ldots, y_n \in Z_p$ and sets the vector $\boldsymbol{s} = \langle s, y_2, \ldots, y_n\rangle^{\mathrm{T}}$, where T denotes the transpose of the vector. It computes a vector $\langle\lambda_1, \lambda_2, \ldots, \lambda_l\rangle^{\mathrm{T}} = \boldsymbol{As}$. Finally, for $\tau = 1$ to $l$, it calculates $C_{4,\tau} = \lambda_\tau - x_{1,\tau}, C_{5,\tau} = -x_{3,\tau}(\rho(\tau) - x_{2,\tau})$. It computes $K' = H(key, C_{3,1}, C_{3,2}, \ldots, C_{3,l+1})$, $C_{4,l+1} = s - x_{1,l+1}$, and $C_{5,l+1} = -x_{3,l+1}(T - x_{2,l+1})$. It outputs a ciphertext $CT_{\mathbb{A},T} = (\mathbb{A}(\boldsymbol{A},\rho), T, m \oplus K', C_0, \{C_{1,\tau}, C_{2,\tau}, C_{3,\tau}, C_{4,\tau}, C_{5,\tau}\}_{\tau\in[l+1]})$.

**Dec**$(CT_{\mathbb{A},T}, dk_{(ID,\omega),T}, pk) \longrightarrow m \setminus \perp$: This algorithm takes in a ciphertext $CT_{\mathbb{A},T} = (\mathbb{A}(\boldsymbol{A},\rho), T, m \oplus K', C_0, \{C_{1,\tau}, C_{2,\tau}, C_{3,\tau}, C_{4,\tau}, C_{5,\tau}\}_{\tau\in[l+1]})$, a decryption key $dk_{(ID,\omega),T} = ((ID,\omega), D_0, D_1, (D_{2,\tau}, D_{3,\tau})_{\tau\in[|\omega|]}, E_0, E_1, E_2, E_3)$ and the public key $pk$. If $\omega$ satisfies the policy $\mathbb{A}(\boldsymbol{A},\rho)$, then it sets $\Gamma = \{\tau \mid \rho(\tau) \in \omega\}$ and computes constants $w_\tau \in Z_p$ such that $\sum_{\tau\in\Gamma} w_\tau \boldsymbol{A}_\tau = \langle 1, 0, \ldots, 0\rangle$ where $\boldsymbol{A}_\tau$ is the $\tau$-th row of matrix $\boldsymbol{A}$, and it denotes $\rho(\tau) = \boldsymbol{A}_i$, where $i$ is the index of attribute $\rho(\tau)$ in $\omega$. It recovers $e(g,g)^{\alpha s}$ by computing

$$\frac{e(C_0, D_0)}{e\left(w^{\sum_{\tau\in\Gamma} C_{4,\tau} w_\tau}, D_1\right) \prod_{\tau\in\Gamma} \left(e(C_{1,\tau}, D_1)\, e\left(C_{2,\tau} u^{C_{5,\tau}}, D_{2,i}\right) e(C_{3,\tau}, D_{3,i})\right)^{w_\tau}} = e(g,g)^{f_{GT}(1)\Delta_{1,I}(0)s}, \quad (9)$$

$$\frac{e(C_0, E_0)}{e(C_{1,l+1} w^{C_{4,l+1}}, E_1)\, e\left(C_{2,l+1} u^{C_{5,l+1}}, E_2\right) e(C_{3,l+1}, E_3)} = e(g,g)^{f_{GT}(T)\Delta_{T,I}(0)s}, \quad (10)$$

$$e(g,g)^{f_{GT}(1)\Delta_{1,I}(0)s} e(g,g)^{f_{GT}(T)\Delta_{T,I}(0)s} = e(g,g)^{\alpha s} = key, \quad (11)$$

$$K' = H(key, C_{3,1}, C_{3,2}, \ldots, C_{3,l+1}),$$
$$m \oplus K' \oplus K' = m. \quad (12)$$

**Rev**$(ID, T, \mathcal{R}, \Lambda) \longrightarrow \mathcal{R}'$: This revocation algorithm takes as inputs an identity $ID$, a revocation time $T$, the revocation list $\mathcal{R}$, and the state $\Lambda = (\mathcal{T}, \mathcal{L}_{ID})$. If $(ID, *) \notin \mathcal{L}_u$, then it outputs $\perp$ if the private key of the pair $(ID,\omega)$ was not generated. Otherwise, it adds $(ID, T)$ to $\mathcal{R}$. It outputs the updated revocation list $\mathcal{R}'$.

### 4.3. Security Proof

**Theorem 7.** *Our scheme is selectively secure under chosen plaintext attacks if the q-type assumption holds. That is, all PPT adversaries have at most a negligible advantage in breaking the R-CP-ABOOE scheme.*

*Proof.* Suppose there exists a PPT adversary $\mathcal{A}$ that breaks the R-CP-ABOOE scheme with a nonnegligible advantage, and then a simulator $\mathcal{S}$ can solve the *q-type* assumption with a nonnegligible advantage by using the given terms of the *q-type* assumption.

**Init**: $\mathcal{S}$ receives the given terms from the *q-type* assumption, a challenge policy $(\boldsymbol{A}^*, \rho^*)$, a challenge time $T^*$ and the revocation list $\mathcal{R}^*$ on the time $T^*$ from $\mathcal{A}$. The challenge matrix $\boldsymbol{A}^*$ is an $l \times n$ matrix and satisfies the restriction $l+1$, $n \le q$, and the map $\rho^*$: $[l] \longrightarrow Z_p$. Let $\mathcal{M}$ be the message space and the size of a message $|\mathcal{M}| = 2^{n_{\mathcal{M}}}$. Let a hash function $H$: $\{0,1\}^* \longrightarrow \{0,1\}^{n_{\mathcal{M}}}$. It sets an $n$-dimensional vector $\boldsymbol{v} = \langle 1, 0, \ldots, 0\rangle$ that is related to the challenge time $T^*$. $\mathcal{S}$ builds a new policy $(\boldsymbol{A}^{**}, \rho^{**})$, where $\rho^{**}$: $[l+1] \longrightarrow \mathcal{U} \cup \{T^*\}$,

$$\boldsymbol{A}^{**} = \begin{pmatrix} \boldsymbol{A}^* \\ \boldsymbol{v} \end{pmatrix},$$
$$\rho^{**}(i) = \{\rho^*(i)\, i \in [l]\, T^* i = l+1. \quad (13)$$

Since $\mathcal{U} \cap \{T^*\} = \varnothing$, we know that, for any attribute set $\omega \subset \mathcal{U}$, $\omega$ satisfies $\boldsymbol{A}^{**}$ if and only if $\omega$ satisfies $\boldsymbol{A}^*$, and for time $T$ that satisfies $\boldsymbol{A}^{**}$ if and only if $T = T^*$.

**Setup**: $\mathcal{S}$ implicitly sets $\alpha = a^{q+1} + \alpha_1$ where $a, q$ are assigned in the assumption and a random exponent $\alpha_1 \in Z_p$ is known only to $\mathcal{S}$. Especially, this way $\alpha$ is well distributed and $a$ is hidden to $\mathcal{A}$ from information theory. Then $\mathcal{S}$ proceeds as follows:

(1) It calls SD.Setup($\mathcal{N}$) to get the full binary tree $\mathcal{T}$ with $2N$ leaves and sets $\mathcal{C}$ to be the collection of all sets $S_{\eta,\theta}$ of $\mathcal{T}$. Let $\mathcal{L}_{ID}$ and $\mathcal{L}_f$ be an empty set respectively. For each user $ID \in \mathcal{R}^*$, it allocates $ID \in \mathcal{N}$ to an unsigned leaf $id$ that belongs to the left subtree of $\mathcal{T}$ and stores $(ID, id)$ in $\mathcal{L}_{ID}$. According to $\mathcal{R}^* \subseteq \mathcal{N}$, it uses $\mathcal{L}_{ID}$ to define the revoked index set $RI^*$. For each $ID \notin \mathcal{R}^*$, it allocates $ID \in \mathcal{N}$ to an unsigned leaf $id$ that belongs to the right subtree of $\mathcal{T}$ and stores $(ID, id)$ in $\mathcal{L}_{ID}$.

(2) It calls $SD.Assign(\mathcal{T}, RI^*)$ to obtain $\bigcup_{id\in RI^*}(PV_{ID})$. Then it sets Fixedsubset($\mathcal{R}^*$) $= \bigcup_{id\in RI^*}(PV_{ID})$. It sets function list $\mathcal{L}_f$ as follows: if $S_{\eta,\theta} \in$ Fixedsubset($\mathcal{R}^*$), it chooses a random value $y \in Z_p$ and stores $(GT = L_\eta \parallel d_\theta, (x = 1, y))$ in $\mathcal{L}_{ID}$. Otherwise, for each $S_{\eta,\theta} \in \mathcal{C} \setminus$ Fixedsubset($\mathcal{R}^*$), it picks random value $y \in Z_p$ and stores $(GT = L_\eta \parallel d_\theta, (T^*, y))$ in $\mathcal{L}_f$ such that $(GT = L_\eta \parallel d_\theta, *) \notin \mathcal{L}_f$. Note that it employs the Lagrange interpolation method to implicitly define $f_{GT}(x)$ by two points $(0, \alpha)$ and $(x, y)$. Next, it sets an empty revocation list $\mathcal{L}_R$ and sets $\Lambda = (\mathcal{T}, \mathcal{L}_{ID})$. It selects random values $v_1, u_1, h_1 \in Z_p$ and gives to $\mathcal{A}$ the following public parameters:

$$
\begin{aligned}
g &= g,\\
w &= g^a,\\
v &= g^{v_1} \cdot g^{a/b_{l+1}} \cdot \prod_{(j,k)\in[l,n]} \left(g^{a^k/b_j}\right)^{\boldsymbol{A}^*_{j,k}},\\
u &= g^{u_1} \cdot g^{a/b^2_{l+1}} \cdot \prod_{(j,k)\in[l,n]} \left(g^{a^k/b_j^2}\right)^{\boldsymbol{A}^*_{j,k}},\\
h &= g^{h_1} \prod_{(j,k)\in[l,n]} \left(g^{a^k/b_j{}^2}\right)^{-\rho*(j)\boldsymbol{A}^*_{j,k}} \cdot g^{-aT^*/b^2_{l+1}},\\
e(g,g)^\alpha &= e\left(g^a, g^{a^q}\right) e(g,g)^{\alpha_1}.
\end{aligned}
\tag{14}
$$

Finally, it publishes public keys $pk = (g, w, v, u, h, e(g,g)^\alpha, H(\cdot))$.

**Phases 1**: $\mathcal{A}$ inquiries adaptively a polynomial number of private key generation, update key and user revocation oracles. If this is a private key query for $(ID, \omega)$, where the attribute set $\omega = (A_1, A_2, \ldots, A_{|\omega|})$, then $\mathcal{S}$ proceeds as follows.

If $\omega$ does not satisfy $(\boldsymbol{A}^*, \rho^*)$ from the pair $(ID, \omega)$, denote $\omega \nvDash (\boldsymbol{A}^*, \rho^*)$, then $\omega \nvDash (\boldsymbol{A}^{**}, \rho^{**})$. It executes the following steps:

(1) It first constructs temporal private key components for the point $(0, \alpha)$ as follows. Since $\omega \nvDash (\boldsymbol{A}^{**}, \rho^{**})$, there exists a vector $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n\rangle \in Z_p{}^n$ such that $w_1 = -1$ and $\boldsymbol{A}_i{}^{**} \cdot \boldsymbol{w} = 0$ for all $i \in I = \{i \mid i \in [l+1] \wedge \rho^{**}(i) \in \omega\}$, and then it picks $r' \in Z_p$ and implicitly sets $r = r' + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1} = r' + \sum_{i\in[n]} w_i a^{q+1-i}$ and computes

$$
\begin{aligned}
P_0 &= g^\alpha w^r = g^{\alpha_1} (g^a)^{r'} \prod_{i=2}^{n} \left(g^{a^{q+2-i}}\right)^{w_i},\\
P_1 &= g^r = g^{r'} \prod_{i\in[n]} \left(g^{a^{q+1-i}}\right)^{w_i}.
\end{aligned}
\tag{15}
$$

In addition, for each attribute $A_\tau \in \omega$, it randomly selects $r_\tau{}' \in Z_p$ and implicitly sets

$$
\begin{aligned}
r_\tau &= r_\tau{}' + r' \sum_{i'\in[l+1],\rho^{**}(i')\notin\omega} \frac{b_{i'}}{A_\tau - \rho^{**}(i')}\\
&\quad + \sum_{(i,i')\in[n,l+1],\rho^{**}(i')\notin\omega} \frac{w_i b_{i'} a^{q+1-i}}{A_\tau - \rho^{**}(i')}.
\end{aligned}
\tag{16}
$$

Then it may compute the terms

$$
\begin{aligned}
P_{2,\tau} &= g^{r_\tau} = g^{r_\tau{}'} \cdot \prod_{i'\in[l+1],\rho^{**}(i')\notin\omega} \left(g^{b_{i'}}\right)^{r'/(A_\tau-\rho^{**}(i'))}\\
&\quad \cdot \prod_{(i,i')\in[n,l+1],\rho^{**}(i')\notin\omega} \left(g^{b_{i'}a^{q+1-i'}}\right)^{w_i/(A_\tau-\rho^{**}(i'))},
\end{aligned}
\tag{17}
$$

$$
\begin{aligned}
P_{3,\tau} &= \left(u^{A_\tau}h\right)^{r_\tau} v^{-r} = \left(u^{A_\tau}h\right)^{r_\tau{}'} \left(P_{2,\tau} g^{-r_\tau{}'}\right)^{u_1 A_\tau + h_1}\\
&\quad \cdot \prod_{\substack{(i',j,k)\in[l+1,l+1,n],\\ \rho^{**}(i')\notin\omega}} \left(g^{b_{i'}a^k b_j^{-2}}\right)^{r' \boldsymbol{A}^{**}_{j,k}(A_\tau-\rho^{**}(j))/(A_\tau-\rho^{**}(i'))}\\
&\quad \cdot \prod_{\substack{(i,i',j,k)\in[n,l+1,l+1,n],\\ \rho^{**}(i')\notin\omega,(j\neq i'\vee i\neq k)}} \left(g^{b_{i'}a^{q+1+k-i}b_j^{-2}}\right)^{(A_\tau-\rho^{**}(j))w_i \boldsymbol{A}^{**}_{j,k}(A_\tau-\rho^{**}(i'))^{-1}} \cdot v^{-r'} \prod_{i\in[n]} \left(g^{a^{q+1-i}}\right)^{-v_1 w_i}\\
&\quad \cdot \prod_{(i,j,k)\in[n,l+1,n],i\neq k} \left(g^{a^{q+1+k-i}b_j^{-1}}\right)^{-w_i \boldsymbol{A}^{**}_{j,k}}.
\end{aligned}
\tag{18}
$$

(2) If $(ID, *) \in \mathcal{L}_{ID}$, then it loads $(ID, id)$ from $\mathcal{L}_{ID}$. Otherwise it allocates $ID \in \mathcal{N}$ to a unique index $id$ that is not assigned previously and stores $(ID, id)$ in $\mathcal{L}_{ID}$. Then it calls $SD.Assign(\mathcal{T}, ID)$ to obtain $PV_{id} = \{S_{\eta,\theta}\}$.

(3) For each $S_{\eta,\theta} \in PV_{id}$, it retrieves $(GT = L_\eta \parallel d_\theta, (x, y))$ from $\mathcal{L}_f$. If $S_{\eta,\theta} \in$ Fixedsubset($\mathcal{R}^*$), it must have $x = 1$ from $\mathcal{L}_f$. It chooses random value $r, r_1, r_2, \ldots, r_{|\omega|} \in Z_p$ and computes a private key by implicitly setting $f_{GT}(1) = y$ as $K_0 = g^y w^r$ and $K_1 = g^r$, and for $\tau = 1$ to $|\omega|$, it computes $K_{2,\tau} = g^{r_\tau}$, $K_{3,\tau} = (u^{A_\tau}h)^{r_\tau} v^{-r}$. Then, it outputs the updated state $\Lambda$ and a private key

$$
\begin{aligned}
sk_{(ID,\omega)} = \Big( &(ID, \omega),\\
&\left\{K_0, K_1, \{K_{2,\tau}, K_{3,\tau}\}_{\tau\in[|\omega|]}\right\}_{S_{\eta,\theta}\in(PV_{ID}\wedge Fixedsubset(\mathcal{R}^*))}\Big).
\end{aligned}
\tag{19}
$$

Otherwise, for each $S_{\eta,\theta} \notin Fixedsubset(\mathcal{R}^*)$, since $x$ is a random value, then it has $x \neq T^*$. Let $I = \{0, T^*\}$, it computes two Lagrange coefficients $\Delta_{0,I}(1)$ and $\Delta_{T^*,I}(1)$. Then it chooses random exponents $r'', r_1{}'', r_2{}'', \cdots, r_k{}'' \in Z_p$ and computes a private key as $K_0 = P_0^{\Delta_{0,I}(1)} (g^y)^{\Delta_{T^*,I}(1)} w^{r''}$ and $K_1 = P_1^{\Delta_{0,I}(1)} g^{r''}$, and for $\tau = 1$ to $|\omega|$, it computes $K_{2,\tau} = P_{2,\tau}^{\Delta_{0,I}(1)} g^{r_\tau{}''}$, and $K_{3,\tau} = P_{3,\tau}^{\Delta_{0,I}(1)} v^{-r''} (u^{A_\tau}h)^{r_\tau{}''}$. Then, it outputs the updated state $\Lambda$ and a private key

$$
\begin{aligned}
sk_{(ID,\omega)} = \Big( &(ID, \omega), \{K_0,\\
&K_1, \{K_{2,\tau}, K_{3,\tau}\}_{\tau\in[|\omega|]}\}_{S_{\eta,\theta}\in PV_{ID}\wedge S_{\eta,\theta}\notin Fixedsubset(\mathcal{R}^*))}\Big).
\end{aligned}
\tag{20}
$$

If $\omega$ satisfies $(\boldsymbol{A}^*, \rho^*)$, it must set $ID \in \mathscr{R}^*$ and executes the following steps:

(1) It loads $(ID, id)$ from $\mathscr{L}_{ID}$. Then it calls $SD.Assign(\mathscr{BT}, ID)$ to obtain $PV_{ID} = \{S_{\eta,\theta}\}$.

(2) For each $S_{\eta,\theta} \in PV_{ID}$, it retrieves $(GT = L_\eta \parallel d_\theta, (1, y))$ from $\mathscr{L}_f$. Since $S_{\eta,\theta} \in Fixedsubset(\mathscr{R}^*)$, it must have $x = 1$ from $\mathscr{L}_f$.. It chooses random value $r, r_1, r_2, \ldots, r_k \in Z_p$ and computes a private key by implicitly setting $f_{GT}(1) = y$ as $K_0 = g^y w^r$, $K_1 = g^r$, and for $\tau = 1$ to $|\omega|$, $K_{2,\tau} = g^{r_\tau}$, and $K_{3,\tau} = (u^{A_\tau} h)^{r_\tau} v^{-r}$. Then, it outputs the updated state $\Lambda$ and a private key

$$sk_{(ID,\omega)} = \left((ID, \omega), \left\{K_0, K_1, \left\{K_{2,\tau}, K_{3,\tau}\right\}_{\tau \in [|\omega|]}\right\}_{S_{\eta,\theta} \in PV_{id}}\right). \quad (21)$$

If $\mathscr{A}$ requests the update key query for time $T$ and $T \neq T^*$, then $T \not\models (\boldsymbol{A}^{**}, \rho^{**})$. $\mathscr{S}$ performs the four steps:

(1) Since $T \not\models (\boldsymbol{A}^{**}, \rho^{**})$, there exists a vector $\boldsymbol{w} = \langle -1, 0, \ldots, 0\rangle^{\mathrm{T}} \in {Z_p}^n$, for all $i \in I = \{i \mid i \in [l+1] \wedge \rho^{**}(i) = T\} = \varnothing$, and then it picks $t' \in Z_p$ and implicitly sets $t = t' - a^q$ and computes

$$\begin{aligned} B_0 &= g^\alpha w^t = g^{\alpha_1} g^{at'}, \\ B_1 &= g^t = g^{t'} g^{-a^q}. \end{aligned} \quad (22)$$

In addition, it randomly selects ${t_1}' \in Z_p$ and implicitly sets

$$t_1 = {t_1}' + t' \sum_{i' \in [l+1]} \frac{b_{i'}}{T - \rho^{**}(i')} - \sum_{i' \in [l+1]} \frac{b_{i'} a^q}{T - \rho^{**}(i')}. \quad (23)$$

Then it may compute the terms

$$\begin{aligned} B_2 &= g^{t_1} \\ &= g^{{t_1}'} \cdot \prod_{i' \in [l+1]} \left(g^{b_{i'}}\right)^{t'/(T - \rho^{**}(i'))} \\ &\quad \cdot \prod_{i' \in [l+1]} \left(g^{b_{i'} a^q}\right)^{-1/(T - \rho^{**}(i'))}, \end{aligned} \quad (24)$$

$$\begin{aligned} B_3 &= \left(u^T h\right)^{t_1} v^{-t} \\ &= \left(u^T h\right)^{{t_1}'} \left(B_2 g^{-{t_1}'}\right)^{u_1 T + h_1} \\ &\quad \cdot \prod_{(i',j,k) \in [l+1, l+1, n]} \left(g^{b_{i'} a^k b_j^{-2}}\right)^{t' \boldsymbol{A}^{**}_{j,k} (T - \rho^{**}(j))/(T - \rho^{**}(i'))} \\ &\quad \cdot \prod_{\substack{(i',j,k) \in [l+1, l+1, n], \\ (j \neq i' \vee k \neq 1)}} \left(g^{b_{i'} a^{q+k} b_j^{-2}}\right)^{-(T - \rho^{**}(j)) \boldsymbol{A}^{**}_{j,k} (T - \rho^{**}(i'))^{-1}} \\ &\quad \cdot v^{-t'} g^{a^q v_1} \cdot \prod_{(j,k) \in [l+1, n], k \neq 1} \left(g^{a^{q+k} b_j^{-1}}\right)^{\boldsymbol{A}^{**}_{j,k}}. \end{aligned} \quad (25)$$

(2) It defines the revocation set on the time $T$ from $\mathscr{R}$, and uses $\mathscr{L}_{ID}$ to define the revoked index set $RI$. It runs SD.Cover$(\mathscr{T}, \mathscr{R})$ to obtain $CV_{RI} = \{S_{\eta,\theta}\}$.

(3) For each $S_{\eta,\theta} \in CV_{RI}$, it retrieves $(GT = L_\eta \parallel d_\theta, (x, y))$ from $\mathscr{L}_f$ and sets $I = \{0, 1\}$. For each $S_{\eta,\theta} \in$ Fixedsubset$(\mathscr{R}^*)$, it chooses random values $t, t_1 \in Z_p$ and computes a time-constrained update key by implicitly setting $f_{GT}(1) = y$ as

$$\begin{aligned} tuk_{T,S_{\eta,\theta}} &= \Big\{U_0 = B_0^{\Delta_{0,I}(T)} \left(g^y\right)^{\Delta_{1,I}(T)} w^t, U_1 \\ &= \left(B_1\right)^{\Delta_{0,I}(T)} g^t, U_2 = \left(B_2\right)^{\Delta_{0,I}(T)} g^{t_1}, U_3 \\ &= B_3^{\Delta_{0,I}(T)} v^{-t} \left(u^T h\right)^{t_1}\Big\}. \end{aligned} \quad (26)$$

For each $S_{\eta,\theta} \notin$ Fixedsubset$(\mathscr{R}^*)$, it sets $I = \{0, T^*\}$ and computes two Lagrange coefficients $\Delta_{0,I}(T)$ and $\Delta_{T^*,I}(T)$. It chooses random values $t'', {t_1}'' \in Z_p$ and computes a time-constrained update key as

$$\begin{aligned} tuk_{T,S_{\eta,\theta}} &= \Big\{U_0 = B_0^{\Delta_{0,I}(T)} \left(g^y\right)^{\Delta_{T^*,I}(T)} w^{t''}, U_1 \\ &= \left(B_1\right)^{\Delta_{0,I}(T)} g^{t''}, U_2 = \left(B_2\right)^{\Delta_{0,I}(T)} g^{{t_1}''}, U_3 \\ &= B_3^{\Delta_{0,I}(T)} v^{-t''} \left(u^T h\right)^{{t_1}''}\Big\}. \end{aligned} \quad (27)$$

(4) Finally, it outputs the updated state $\Lambda$ and an update key

$$uk_{T,\mathscr{R}} = \left(CV_{RI}, \left\{tuk_{T,S_{\eta,\theta}}\right\}_{S_{\eta,\theta} \in CV_{RI}}\right). \quad (28)$$

If $T = T^*$, it executes the following three steps:

(1) It sets $\mathscr{R} = \mathscr{R}^*$ and obtains the revoked index set $RI^*$ of the revocation list $\mathscr{R}^*$ by $\mathscr{L}_{ID}$. It does not run SD.Cover$(\mathscr{T}, \mathscr{R}^*)$ to obtain $CV_{RI^*}$ but assigns all the nonrevoked users to the right subtree of $\mathscr{T}$; *i.e.*, $CV_{RI^*}$ can be described as the set $\{S_{\eta,\theta} \mid \eta$ is the root node of the right subtree of $\mathscr{T}$, $\theta$ is the descendant node of $\eta\}$. Thus, $CV_{RI^*} \cap$ Fixedsubset$(\mathscr{R}^*) = \varnothing$.

(2) For each $S_{\eta,\theta} \in CV_{RI^*}$, it sets $GT = L_\eta \parallel d_\theta$ and retrieves $(GT, (T^*, y))$ from $\mathscr{L}_f$. Then it chooses random values $t, t_1 \in Z_p$ and computes a time-constrained update key by implicitly setting $f_{GT}(T^*) = y$ as

$$\begin{aligned} &tuk_{T,S_{\eta,\theta}} \\ &= \left\{U_0 = g^y w^t, U_1 = g^t, U_2 = g^{t_1}, U_3 = \left(u^{T^*} h\right)^{t_1} v^{-t}\right\}. \end{aligned} \quad (29)$$

(3) Finally, it outputs the updated state $\Lambda$ and an update key

$$uk_{T,\mathscr{R}} = \left(CV_{RI}, \left\{tuk_{T,S_{\eta,\theta}}\right\}_{S_{\eta,\theta} \in CV_{RI}}\right). \quad (30)$$

**Challenge**: The adversary $\mathscr{A}$ submits a pair of messages $(m_0, m_1)$ with the same length to the simulator $\mathscr{S}$. Then $\mathscr{S}$ flips a random bit $c \in \{0, 1\}$ and constructs: $key = F \cdot e(g, g^s)^{\alpha_1}$, $C_0 = g^s$, where $F$ is the challenge term and $g^s$ is the appropriate term of the assumption.

$\mathcal{S}$ implicitly sets a vector $\boldsymbol{y} = (s, sa + y_2, \ldots, sa^{q+1} y_n)$, where $y_2, \ldots, y_n$ are chosen uniformly at random from $Z_p$. Therefore, the secret $s$ and the vector $\boldsymbol{y}$ are uniformly distributed. Especially $s$ is hidden for $\mathcal{A}$ from information theory. Let $\boldsymbol{\lambda} = \boldsymbol{A}^* \boldsymbol{y}^{\mathrm{T}}$. For each row $\tau \in [l]$, it implicitly sets

$$\lambda_\tau = \sum_{i \in [n]} \boldsymbol{A}^*_{\tau,i} sa^{i-1} + \sum_{i \in [n]} \boldsymbol{A}^*_{\tau,i} y_i = \sum_{i \in [n]} \boldsymbol{A}^*_{\tau,i} sa^{i-1} + \lambda'_\tau. \quad (31)$$

Let the terms $\lambda'_\tau$ be known to $\mathcal{S}$. For each row of $\boldsymbol{A}^*_\tau$, $\mathcal{S}$ implicitly sets $t_\tau = -sb_\tau$. Since $b_\tau$'s are hidden for $\mathcal{A}$ from information theory, the terms $t_\tau$ are uniformly distributed as well. $\mathcal{S}$ also picks random values $x'_{1,\tau}, x'_{2,\tau}, x'_{3,\tau} \in Z_p$ and implicitly sets $x_{1,\tau} = \lambda_\tau - x'_{1,\tau}$, $x_{2,\tau} = \rho(\tau)^* - x'_{2,\tau} \cdot b_\tau^{-1}$, and $x_{3,\tau} = -sb_\tau$. As a result, $\mathcal{A}$ computes

$$C_{1,\tau} = w^{x_{1,\tau}} v^{x_{3,\tau}} = w^{\lambda_\tau} \left(g^{sb_\tau}\right)^{-\nu_1} \cdot w^{-x'_{1,\tau}} \prod_{(j,k) \in [l+1,n] \wedge j \neq \tau} \left(g^{sa^k b_\tau b_j^{-1}}\right)^{-\boldsymbol{A}^{**}_{j,k}}, \quad (32)$$

$$C_{2,\tau} = \left(u^{x_{2,\tau}} h\right)^{-x_{3,\tau}} = u^{-x'_{2,\tau}} \left(g^{sb_\tau}\right)^{-(u_1 \rho^{**}(\tau) + h_1)} \cdot \prod_{(j,k) \in [l,n] \wedge j \neq \tau} \left(g^{sa^k b_\tau b_j^{-2}}\right)^{-\boldsymbol{A}^{**}_{j,k} (\rho^{**}(\tau) - \rho^{**}(j))}, \quad (33)$$

$$C_{3,\tau} = g^{x_{3,\tau}} = g^{-sb_\tau},$$

$$C_{4,\tau} = \lambda_\tau - x_{1,\tau} = x'_{1,\tau}, \quad (34)$$

$$C_{5,\tau} = -x_{3,\tau} \left(\rho^{**}(\tau) - x_{2,\tau}\right) = x'_{2,\tau},$$

For $\tau = l + 1$, $\mathcal{S}$ implicitly sets $t_{l+1} = -sb_{l+1}$ and also picks random values $x'_{1,l+1}, x'_{2,l+1} \in Z_p$ and implicitly sets $x_{1,l+1} = s - x'_{1,l+1}$, $x_{2,l+1} = T^* - x'_{2,l+1}$, and $x_{3,l+1} = -sb_{l+1}$. As a result, $\mathcal{A}$ computes

$$C_{1,l+1} = w^{x_{1,l+1}} v^{x_{3,l+1}} = \left(g^{sb_{l+1}}\right)^{-\nu_1} \prod_{(j,k) \in [l+1,n] \wedge j \neq \tau} \left(g^{sa^k b_{l+1} b_j^{-1}}\right)^{-\boldsymbol{A}^{**}_{j,k}}, \quad (35)$$

$$C_{2,l+1} = \left(u^{x_{2,l+1}} h\right)^{-x_{3,l+1}} = \left(g^{sb_{l+1}}\right)^{-(u_1 T^* + h_1)} \prod_{(j,k) \in [l+1,n]} \left(g^{sa^k b_{l+1} b_j^{-2}}\right)^{-\boldsymbol{A}^{**}_{j,k} (T^* - \rho^{**}(j))}, \quad (36)$$

$$C_{3,l+1} = g^{x_{3,l+1}} = g^{-sb_{l+1}},$$

$$C_{4,l+1} = s - x_{1,l+1} = x'_{1,l+1}, \quad (37)$$

$$C_{5,l+1} = -x_{3,l+1} \left(T^* - x_{2,l+1}\right) = x'_{2,l+1},$$

$$K' = H\left(key, C_{3,1}, C_{3,2}, \ldots, C_{3,l+1}\right). \quad (38)$$

Finally, $\mathcal{S}$ submits the ciphertext $CT^* = \{(\boldsymbol{A}^*, \rho^*), T^*, C_0 = m_c \oplus K', \{C_{1,\tau}, C_{2,\tau}, C_{3,\tau}, C_{4,\tau}, C_{5,\tau}\}_{\tau \in [l+1]}\}$ to $\mathcal{A}$.

**Phase 2**: it is the same as Phase 1.

**Guess:** $\mathcal{A}$ outputs a guess $c'$ for the challenge ciphertext $CT^*$. If $c' = c$, $\mathcal{S}$ outputs 0, it means that the challenge term is $F = e(g, g)^{-sa^{q+1}}$. Otherwise, it outputs 1.

If $F = e(g, g)^{-sa^{q+1}}$, $\mathcal{A}$ played the real security game, since $key = F \cdot e(g, g)^{\alpha s}$, $K' = H(key, C_{3,1}, C_{3,2}, \ldots, C_{3,l+1})$, and $C_0 = m_c \oplus K'$. Otherwise, if $F$ is a random value of $G_T$, then the advantage of $\mathcal{A}$ is 0. Therefore, if $\mathcal{A}$ wins the above security game with a nonnegligible advantage, then $\mathcal{S}$ can solve the *q-type* assumption with a nonnegligible advantage by using the given terms of the *q- type* assumption. □

## 5. Performance Analysis

This section elaborates the comparisons between our R-CP-ABOOE and some related ABE schemes on the functionalities and efficiency respects. We summarized the comparison results in Table 1, where $m_c$, M and E, respectively, denote a modular operation in $Z_p$, a multiplication, and an exponentiation in the groups $G$ or $G_T$. Let $l$ be the number of rows in the matrix $\boldsymbol{A}$ and $r$ be the number of revoked users in $\mathcal{R}$. Let $|\omega|$, $N$, $|U|$, $|G|$, $|G_T|$, and $|\mathcal{M}|$ represent the number of attributes in the collection $\omega$, the total number of users, the number of all attributes, the size of an element in $G$, the size of an element in $G_T$, and the size of a message in the space $\mathcal{M}$, respectively.

In Table 1, the revocable ABE schemes [5, 17, 32] and our R-CP-ABOOE scheme support user revocation. However, the real (online) encryption algorithms in the schemes [5, 17, 32] must perform many exponentiation operations, which is unsuitable for mobile devices with limited-computation power. Fortunately, in our scheme, first the offline-encryption phase can be executed by high performance computer in a trusted environment. Then our real (or online) encryption only requires $3l + 2$ modular operations and does not require any exponentiation. Modular operation is much faster than the exponentiation operation. Thus, our real encryption algorithm is the fastest among these schemes [5, 17, 32].

In addition, compared with the indirectly revocable ABE [5], our scheme may significantly reduce the key-update cost and the size of update key from $O(r \log(N/r))$ to $O(r)$, which is important since the update keys should be periodically broadcasted to all nonrevoked users. Compared with the directly revocable ABE [17, 32] that cannot be integrated with the online/offline technology, our encryption algorithm and ciphertext size are not related to the number of revoked users. Therefore, our scheme has less encryption overhead and shorter ciphertext length than the directly revocable ABE [17, 32], which is suitable for data owners with resource-limited devices. Although our scheme need store additional update key and small amount of offline ciphertexts, current mobile devices are perfectly capable. The online/offline ABE scheme [9] cannot provide the user revocation, whereas our scheme achieves the user revocation mechanism without excessive computational and storage costs. In general, our R-CP-ABOOE scheme is the first CP-ABE scheme, which can simultaneously support the user revocation and the online/offline encryption mode, and it has desirable features of very little encryption overhead for mobile device and less

Table 1: Comparisons of the related attribute-based encryption schemes.

| Schemes | Online-Enc cost | Key-update cost | Ciphertext size | Update-key size | Rev |
|---|---|---|---|---|---|
| [5] | $(\|\omega\| + 2)$E | $r\log(N/r)$E | $(\|\omega\| + 1)\|G\| + \|G_T\|$ | $r\log(N/r)\|G\|$ | √ |
| [17] | $(2l + r\log(N/r) + 3)$E | - | $(2 + r\log(N/r) + l)\|G\| + \|G_T\|$ | - | √ |
| [32] | $(80\|\omega\| + 160r - 48)$E | - | $(16\|\omega\| + 64r - 27)\|G\| + \|G_T\|$ | - | √ |
| [9] | $3lm_c$ | - | $(3l + 1)\|G\| + 2l\|Z_p\| + \|\mathcal{M}\|$ | - | × |
| Ours | $(3l + 2)m_c$ | $(14r\text{-}7)$E | $(3l + 4)\|G\| + 2(1 + l)\|Z_p\| + \|\mathcal{M}\|$ | $(8r - 4)\|G\|$ | √ |

the number of group elements in a ciphertext and update key. Performance analysis shows that our scheme greatly improves the key-update efficiency for the trusted party and the encryption efficiency for mobile devices.

## 6. Conclusions

In this work, we deal with the key-update efficiency and the encryption efficiency issues in revocable CP-ABE systems and propose an efficient ciphertext-policy attribute-based online/offline encryption with user revocation (R-CP-ABOOE), which is proven to be selectively secure under the $q$-type assumption. Our R-CP-ABOOE scheme simultaneously supports user revocation and online/offline encryption mode and significantly improves the key-update efficiency from $O(r\log(N/r)$ to $O(r)$. Moreover, our scheme has desirable features of very little encryption overhead for mobile device and less group elements in the update key. Performance analysis shows that our scheme greatly improves the key-update efficiency for the trusted party and the encryption efficiency for mobile devices. Furthermore, we can employ the outsourcing decryption technology [10, 16, 33] to reduce the decryption cost.

## Data Availability

"No data were used to support this study."

## Conflicts of Interest

The three authors confirm that they have no conflicts of interest.

## Acknowledgments

This work is supported by Jiangsu Overseas Visiting Scholar Program for University Prominent Young and Middle-aged Teachers and Presidents, the National Natural Science Foundation of China (Nos. 61402244, 11371207, and 61762044), Nantong City Application Basic Research Project (No. GY12017024), and the Zhejiang Natural Science Foundation (LY15F020010).

## References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology – EUROCRYPT*, pp. 457–473, Aarhus, 2005.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, November 2006.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, Berkeley, May 2007.

[4] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the ACM SIGSAC conference on Computer and communications security, ACM '13*, pp. 463–474, 2013.

[5] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proceedings of the 14th ACM Conference on Computer and Communications Security. Cryptology ePrint Archive*, 2008, https://eprint.iacr.org/2012/052.pdf.

[6] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Advances in Cryptology—CRYPTO*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 41–62, Springer, Berlin, Germany, 2001.

[7] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," in *Proceedings of the 12th IMA International Conference Cryptography and Coding*, pp. 278–300, 2009.

[8] K. Lee, D. H. Lee, and J. H. Park, "Efficient revocable identity-based encryption via subset difference methods," *Designs, Codes and Cryptography*, vol. 85, no. 1, pp. 39–76, 2017.

[9] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-key cryptography (PKC)*, vol. 8383 of *Lecture Notes in Comput. Sci.*, pp. 293–310, Springer, Heidelberg, 2014.

[10] Z.-J. Wang, H.-Y. Ma, and J.-H. Wang, "Attribute-based online/offline encryption with outsourcing decryption," *Journal of Information Science and Engineering*, vol. 32, no. 6, pp. 1595–1611, 2016.

[11] P. Datta, R. Dutta, and S. Mukhopadhyay, "Fully Secure Online/Offline Predicate and Attribute-Based Encryption," in *Information Security Practice and Experience*, vol. 9065 of *Lecture Notes in Computer Science*, pp. 331–345, Springer International Publishing, Berlin, Germany, 2015.

[12] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.

[13] J. G. Li, W. Yao, Y. C. Zhang, H. L. Qian, and J. G. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.

[14] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full Verifiability for Outsourced Decryption in Attribute Based Encryption," *IEEE Transactions on Services Computing*, 2017.

[15] J. Li, X. Y. Huang, J. W. Li, X. F. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2201–2210, 2014.

[16] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2119–2130, 2015.

[17] Z. Liu, S. Duan, P. Zhou et al., "Traceable-then-revocable ciphertext-policy attribute-based encryption scheme," *Future Generation Computer Systems*, 2017, https://doi.org/10.1016/j.future.2017.09.045.

[18] H. Y. Ma and G. S. Zeng, "An attribute-based encryption scheme for traitor tracing and revoking," *Chinese Journal of Computers. Jisuanji Xuebao*, vol. 35, no. 9, pp. 1845–1855, 2012.

[19] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing," *Future Generation Computer Systems*, vol. 78, pp. 720–729, 2018.

[20] H. Ma, G. Zeng, Z. Bao, J. Chen, J. Wang, and Z. Wang, "Attribute-based encryption scheme resilient against continuous auxiliary-inputs leakage," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 53, no. 8, pp. 1867–1878, 2016.

[21] J. Li, Q. Yu, Y. Zhang, and J. Shen, "Key-Policy Attribute-Based Encryption against Continual Auxiliary Input Leakag," *Information Sciences*, vol. 470, pp. 175–188, 2019.

[22] S. Even, O. Goldreich, S. Micali et al., "On-line/off-line digital signatures," *Journal of Cryptology*, vol. 9, no. 1, pp. 35–67, 1996.

[23] F. Guo C, Y. Mu, and Z. Chen, "Identity-based online/offline encryption," in *Financial Cryptography*, pp. 247–261, Springer, Berlin, 2008.

[24] S. S. Chow, J. K. Liu, and J. Zhou, "Identity-based online/offline key encapsulation and encryption," in *Proceedings of the ASI-ACCS*, pp. 52–60, Springer, Berlin, 2011.

[25] J. K. Liu and J. Zhou, "An efficient identity-based online/offline encryption scheme," in *Proceedings of ACNS*, pp. 156–167, Springer, Berlin, 2009.

[26] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Pairing-Based Cryptography (Pairing '09)*, pp. 248–265, Springer, Berlin, Germany, 2009.

[27] Q. Li, H. Xiong, and F. Zhang, "Broadcast revocation scheme in composite-order bilinear group and its application to attribute-based encryption," *International Journal of Security and Networks*, vol. 8, no. 1, pp. 1–12, 2013.

[28] Y. Shi, Q. Zheng, J. Liu, and Z. Han, "Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation," *Information Sciences*, vol. 295, pp. 221–231, 2015.

[29] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Advances in cryptology-CRYPTO*, vol. 7417 of *Lecture Notes in Comput. Sci.*, pp. 199–217, Springer, Berlin, Germany, 2012.

[30] T. Ishiguro, S. Kiyomoto, and Y. Miyake, "A key-revocable attribute-based encryption for mobile cloud environments," in *Proceedings of IEEE Symposium on Security Cryptography*, pp. 1–11, Iceland, 2015.

[31] H. Cui, R. Deng H, X. Ding et al., "Attribute-based encryption with granular revocation," in *Proceedings of the Conference on Security and Privacy in Communication Systems*, pp. 165–181, Springer, Berlin, Germany, 2016.

[32] P. Datta, R. Dutta, and S. Mukhopadhyay, "Adaptively secure unrestricted attribute-based encryption with subset difference revocation in bilinear groups of prime order," in *Progress in cryptology-AFRICACRYPT*, vol. 9646 of *Lecture Notes in Comput. Sci.*, pp. 325–345, Springer, Berlin, Germany, 2016.

[33] B. Qin, R. H. Deng, S. Liu, and S. Ma, "Attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1384–1393, 2015.