Ciphertext-Policy Attribute-Based Encrypted Data Equality Test and Classification[⋆]

Yuzhao Cui[1], Qiong Huang[1⋆⋆], Jianye Huang[1], Hongbo Li[1], and Guomin Yang[2]

[1] College of Mathematics and Informatics, South China Agricultural University,
Guangzhou 510642, China

[2] School of Computing and Information Technology, University of Wollongong,
Wollongong, NSW 2522, Australia

Keywords: Attribute-Based Encryption; Authorization; Classification; Equality Test; Public Key Encryption

Abstract. Thanks to the ease of access and low expenses, it is now popular for people to store data in cloud servers. To protect sensitive data from being leaked to the outside, people usually encrypt the data in the cloud. However, management of these encrypted data becomes a challenging problem, e.g. data classification. Besides, how to selectively share data with other users is also an important and interesting problem in cloud storage. In this paper, we focus on *ciphertext-policy attribute based encryption with equality test* (CP-ABEET). People can use CP-ABEET to implement not only flexible authorization for the access to encrypted data, but also efficient data label classification, i.e. test of whether two encrypted data contain the same message. We construct an efficient CP-ABEET scheme, and prove its security based on a reasonable number-theoretic assumption. Compared with the only existing CP-ABEET scheme, our construction is more efficient in key generation, and has shorter attribute-related secret keys and better security.

Keywords. Attribute-Based Encryption, Authorization, Classification, Equality Test, Public Key Encryption

1 Introduction

In the recent years, cloud computing technology has been well studied and is becoming more and more popular in our daily life. Storage as a service (SaaS) is an important component of cloud computing, and is widely used nowadays.

[⋆] This work is supported by National Natural Science Foundation of China (Nos. 61872152, 61472146), Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2014A030306021), Guangdong Program for Special Support of Top-notch Young Professionals (No. 2015TQ01X796), and Pearl River Nova Program of Guangzhou (No. 201610010037).

[⋆⋆] Qiong Huang is the corresponding author. Email: `qhuang@scau.edu.cn`.

People store large amounts of data in cloud to decrease the local storage burden. However, data privacy is under threat because of the openness of public clouds. To protect the user privacy, people are inclined to encrypt the data in. However, encryption would make data process become difficult. After encryption, data structures are usually hidden so that logical operations and other computation operations could not be applied any more. For example, users cannot directly search over encrypted files stored in the cloud. A naive method is to download all the encrypted files from the cloud, decrypt them and then use traditional methods to search over the plaintext files. Although in this way the data can be searched, but it is cumbersome and requires a large computation and storage cost, as well as a high requirement on the bandwidth, which is impractical.

When a company outsources the storage of a large volume of encrypted data to the cloud, the management of these data becomes a complex problem. It is necessary to label the data and classify them into different categories. We need a mechanism to efficiently divide all the encrypted data into groups according to data attributes. Thus, it is interesting and important to study the problem of *label classification* of encrypted data.

On the other hand, access control of (encrypted) data in a company is also a key issue. Each employee in the company has different attributes. Different employees are provided different privileges to access different part of these data. The access privileges are usually authorized according their attributes. For example, tax data of users in the company should only be accessed by an employee associated with attributes "*Department of Finance*" and "*Tax Officer*" or an employee with attributes "*Finance Manager*" , but cannot be accessed by an employee with attributes "*Department of R&D*" and "*Programmer*". How to design a mechanism to effectively manage the access control for all users in the company is thus an important problem.

Yang et al. [19] introduced the primitive of public key encryption with equality test (PKEET) which allows anyone to efficiently test if two ciphertexts w.r.t different public keys contain the same message without decryption. This special property makes it suitable for implementing label classification. On the other hand, ABE [13], especially Ciphertext-policy ABE (CP-ABE) [3], is a good utility for access control. Ciphertext-policy ABE (CP-ABE) is a variant of ABE, in which an access policy is embedded in a ciphertext while each user a formal set of attributes is embedded in a secret key of each user. A user can decrypt ciphertexts only if the embedded access policy can be satisfied by the user's attributes. CP-ABE with equality test (CP-ABEET) [15], integrating the advantage of PKEET and CP-ABE, can be used to classify encrypted data, and in the meanwhile, implement flexible control of access to the encrypted data in cloud. Roughly, CP-ABEET works as follows.

(**Controlled Classification of Encrypted Data**). First of all, attribute distribution is under control of the regulatory agency of the company. There are some data managers in the company, and each of them is in charge of the management and maintenance of different part of the company's data. Authorization privileges for data classification are represented by attribute sets. Employees of

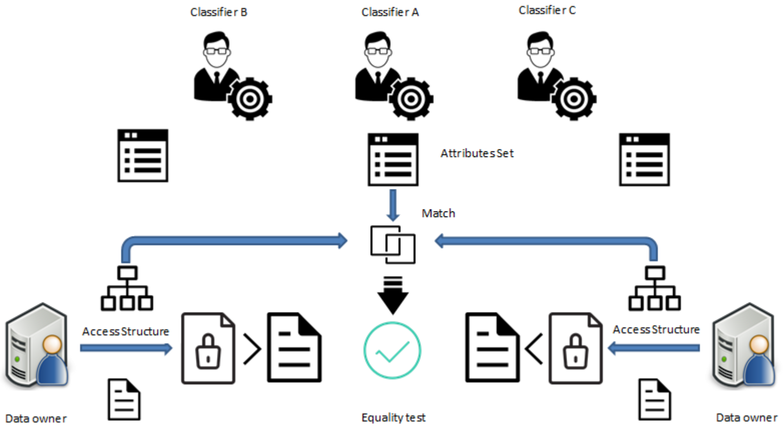

Fig. 1. System architecture of CP-ABEET

the company, including the data managers, receive attribute-related secret keys from the regulatory agency. Before uploading to the cloud server, data owner encrypts its data using a block cipher, e.g. AES. It also sets a label for the data, an access policy specifying which data manager(s) can access the data and its label, and encrypts the label using CP-ABEET encryption algorithm. The data classifier obtains trapdoors from different data managers before classifying the data. Given the Tds, it is then able to access the data as long as the attribute sets embedded in the Tds satisfy the access policies embedded in the ciphertexts, and classify these data according to their labels using the test function of CP-ABEET. If two pieces of data are attached with the same label, they will be put into the same category. Besides, the data managers are able to decrypt the ciphertexts to recover the original data (as well as the label) using the decryption function of CP-ABEET. Figure 1 shows the system architecture of CP-ABEET.

(Our Contributions). Although the CP-ABEET scheme proposed by Wang et al. [15] is concrete, it is complex and requires a large computation for the server, which makes it not suitable for practical use, e.g. label classification. Besides, their scheme only achieves CPA-type indistinguishability when the adversary is not given trapdoors, which does not capture some real-life attacks.

In this work, we propose a novel construction of CP-ABEET, and prove that our construction achieves one-wayness if the adversary is given test trapdoors, and achieves indistinguishability if the adversary is not given trapdoors. Compared with the only known CP-ABEET scheme [15], our scheme is more computationally efficient in key generation and has shorter attribute-related secret keys. Besides, our scheme is CCA-type secure, while Wang et al.'s scheme [15] is only CPA-type secure. The price of our scheme is that we need a stronger number-theoretic assumption and more computational costs in decryption and test algorithms. Table 1 (page 17) provides a detailed comparison between our scheme and some related schemes, e.g. [9, 15, 20].

Paper Organization. We review related works in Sect. 2. Then in Sect. 3 we introduce the preliminaries which are needed for our construction. In Sect. 4 we define CP-ABEET and its security models. The construction of CP-ABEET is given in 5. The security analysis of CP-ABEET is provided in Sect. 6. We compare our scheme with some related schemes in Sect. 7. Finally, we conclude the paper in Sect. 8.

2 Related Works

(ABE): Sahai et al. initialized the study of fuzzy identity-based encryption, the embryo of ABE, in 2005 [13]. Goyal et al. and Bethencourt et al. presented the first key-policy ABE (KP-ABE) scheme [5] and the first ciphertext-policy ABE (CP-ABE) scheme [3], respectively. There are many follow-ups, e.g. [1,5,7], [4,16], and etc. Based on these ABE schemes, researchers proposed more complex and flexible ABE schemes. However, there are still some issues with these ABE schemes, among which the efficiency is a major one, e.g. ciphertext size and decryption cost. Green et al. [6] gave a new method of efficiently and securely outsourcing the decryption of ABE ciphertexts to a third-party server, which reduces the overhead of users significantly. To further enhance user privacy, in some ABE schemes [8,12], the access policy is *hidden* so that an adversary cannot learn anything about the policy from the ciphertexts.

(PKEET): Yang et al. defined the concept of PKEET in 2010 [19] and presented a concrete construction which allows to test if two ciphertexts decrypt to a common result without decryption. However, since any entity is able to run the test in their scheme, an adversary may learn some information about the message from ciphertexts. An authorization mechanism is thus needed to control the access to the test function. Fine-grained authorization policy PKEET (FG-PKEET) was proposed to implement the accurate authorization in PKEET [14], in which only the authorized users can perform the equality test.Public key encryption with delegated equality test (PKE-DET) was proposed to optimize the authorization mechanism [11], in which only the delegated party can run the test. Furthermore, a PKEET scheme with flexible authorization was proposed in [10], which further refines the authorization into four types.

(IBEET): As PKEET works in PKI, the management of certificates is complex. Ma et al. defined the concept of *identity-based encryption with outsourced equlity test* (IBEET) [9] to solve the aforementioned issue. A user in IBEET computes a trapdoor Td using the secret key w.r.t. its identity and gives Td to the server for equality test. Releasing the trapdoor indicates that the user delegates out its equality testing capability. IBEET can be applied in encrypted database system in which the server hosts the database and users could run equality test on encrypted records. Considering the threat of curious database server, Wu et al. [18] presented an IBEET scheme against insider attacks. Besides, to improve the efficiency, Wu et al. reduced the use of *HashToPoint* function in their another IBEET scheme [17], which is time costly. It is restricted in their scheme that only particular keywords can be tested, in order to improve the security level of their scheme.

(ABEET): There are not many schemes focusing on *attribute based encryption with equality test* (ABEET). Considering the relationship between ABE and IBE, Zhu et al. proposed the first *key-policy ABEET* (KP-ABEET) scheme [20], which provides more flexible authorization than previous works on PKEET and IBEET. Wang et al. presented a *ciphertext-policy ABEET* (CP-ABEET) [15] very recently. Both of the schemes are complex and suffer from high computational complexity.

3 Preliminaries

In this part we give a brief review of some basic definitions which are necessary for our construction of CP-ABEET.

The following definitions of access structure and linear secret sharing scheme (LSSS) are adapted from [16] and [2], respectively.

Definition 1 (Access Structure, AC [16]). *Let $\mathbb{P} = \{P_i\}_{i=1}^n$ be a set of n parties, and $\mathbb{A}$ be a subset of $2^{\mathbb{P}}$. We say $\mathbb{A}$ is* monotone *if $\forall S_1, S_2, (S_1 \in \mathbb{A}) \wedge (S_1 \subseteq S_2) \rightarrow (S_2 \in \mathbb{A})$. A monotone collection $\mathbb{A} \subseteq 2^{\mathbb{P}} \backslash \{\emptyset\}$ is called a* monotone access structure*. Sets in $\mathbb{A}$ are* authorized*, and those outside of $\mathbb{A}$ are* unauthorized*.*

Definition 2 (LSSS [2]). *We say a secret sharing scheme Π over a set of parties $\mathbb{P}$ is* linear *(over $\mathbb{Z}_p$) if the following two conditions hold.*

1. *For each party in $\mathbb{P}$, the secret shares form a vector over $\mathbb{Z}_p$.*
2. *There exists a share generating matrix M of size $\ell \times n$. We use a map $\rho(\cdot)$ to connect each row of M with its corresponding party in $\mathbb{P}$. Let $s \in \mathbb{Z}_p$ be the secret to be shared, and $r_2, \cdots, r_n$ be random elements of $\mathbb{Z}_p$. The vector Mv, where $v = (s, r_2, \cdots, r_n)$, contains the shares of s according to Π, and $(Mv)_i$ is the share belonging to party $\rho(i)$.*

There is an efficient algorithm which can find a set of constants $\{w_i\}$ for recovering the secret s, e.g. $\sum_{i \in I} w_i \lambda_i = s$, where I is the set of indices of parties in an authorized set and $\{\lambda_i\}$ are valid shares of s generated by Π [2]. This is known as the *linear reconstruction* property. Same as [16], we use $(1, 0, \cdots, 0)$ as the target vector for LSSS. For any satisfying set of rows I in M, there exists a vector w s.t. $w \cdot (1, 0, \cdots, 0) = -1$ and $\forall i \in I, w \cdot M_i = 0$.

Bilinear Pairing: Given cyclic groups $\mathbb{G}, \mathbb{G}_T$ of prime order p and a generator g of $\mathbb{G}$, we say $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear pairing if (1) $\forall g_1, g_2 \in \mathbb{G}$, $\forall x, y \in \mathbb{Z}_p$, $e({g_1}^x, {g_2}^y) = e(g_1, g_2)^{xy}$; (2) $e(g, g)$ generates $\mathbb{G}_T$; and (3) $\forall g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2)$ can be computed in polynomial time.

(Decisional q-Parallel BDHE Assumption [16]): Suppose $\mathbb{G}$ is a group of prime order p, and g is a generator. Choose at random $s, a, b_1, \cdots, b_q \in \mathbb{Z}_p$. Denote by

$$\begin{gathered}
\mathbf{y} := \Big(g, g^s, g^a, \cdots, g^{(a^q)}, , g^{(a^{q+2})}, \cdots, g^{(a^{2q})}; \\
\forall 1 \le j \le q, g^{s \cdot b_j}, \\
g^{a/b_j}, \cdots, g^{(a^q/b_j)}, g^{(a^{q+2}/b_j)}, \cdots, g^{(a^{2q}/b_j)}; \\
\forall 1 \le j, k \le q, k \ne j, g^{(a \cdot s \cdot b_k/b_j)}, \cdots, g^{(a^q \cdot s \cdot b_k/b_j)} \Big).
\end{gathered}$$

Decisional q-parallel Bilinear Diffie-Hellman Exponent problem (BDHE) is that given $\mathbf{y}$, the adversary could not distinguish $e(g,g)^{a^{q+1}s}$ from a random element $R \in \mathbb{G}_T$.

Definition 3 (Decisional q-parallel BDHE Assumption). *We say that the Decisional q-parallel BDHE assumption holds if for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\mathbf{y}, T = e(g,g)^{a^{q+1}s}) = 0] - \Pr[\mathcal{A}(\mathbf{y}, T = R) = 0]|$$

is negligible.

4 CP-ABEET

4.1 Definition

Definition 4 (CP-ABEET). *A CP-ABEET scheme consists of PPT algorithms* (**Setup**, **Encrypt**, **KeyGen**, **Trapdoor**, **Test**, **Decrypt**)*, as below.*

- **Setup***: Given a security parameter 1^k and the maximal number N of attributes in the system, it outputs a master public/secret key pair* $(\mathsf{Mpk}, \mathsf{Msk})$*.*
- **Encrypt***: Given* Mpk*, an access structure (M, ρ) and a message m as input, it outputs a ciphertext* Ct*.*
- **KeyGen***: Given* Msk *and an attribute set S, it outputs a secret key* Sk_S *for S.*
- **Trapdoor***: Given* Msk *and an attribute set S, it outputs a trapdoor* Td_S *for S.*
- **Test***: Given* $\mathsf{Ct}_A, \mathsf{Td}_A$ *of user A and* $\mathsf{Ct}_B, \mathsf{Td}_B$ *of user B, it outputs 1 if* Ct_A *and* Ct_B *contain the same plaintext, and 0 otherwise.*
- **Decrypt***: Given* Ct *and* Sk_S*, it outputs a message m or a failure symbol $\perp$.*

Correctness requires that (1) an honestly generated ciphertext Ct could be correctly decrypted by a secret key Sk_S if S satisfies the access structure embedded in Ct; and (2) honestly generated ciphertexts of the same message and honestly generated trapdoors could pass the equality test, as long as attributes in trapdoors satisfy the access structures in ciphertexts, respectively.

4.2 Security Models

We consider two security properties of CP-ABEET. Let $\mathcal{A}$ be an adversary. If $\mathcal{A}$ is given the trapdoor, we require that it cannot recover the message from a given ciphertext. Otherwise, we require that it cannot distinguish a given ciphertext encapsulates which message. Below we formally define the security properties by two games, where $\mathcal{C}$ is a challenger.

One-wayness against chosen access structure and chosen ciphertext attacks (OW-CAS-CCA):

1. **Setup**. $\mathcal{C}$ prepares a master key pair $(\mathsf{Mpk}, \mathsf{Msk})$ and sends Mpk to $\mathcal{A}$.
2. **Query Phase 1**. $\mathcal{A}$ adaptively issues queries for polynomially many times.
   - *ExtractQuery.* $\mathcal{A}$ submits a set S of attributes, and is returned a corresponding secret key Sk_S.
   - *TrapdoorQuery.* $\mathcal{A}$ submits a set S of attributes, and is returned a corresponding trapdoor Td_S.
   - *DecryptionQuery.* $\mathcal{A}$ submits a ciphertext Ct and a set S of attributes, and is returned the corresponding decryption result.
3. **Challenge Phase**. $\mathcal{A}$ submits (M^*, ρ^*) as the challenge access structure such that $\mathcal{A}$ did not ask an Extract query on input any S satisfying (M^*, ρ^*) in **Query Phase 1**. $\mathcal{C}$ chooses a random message m^*, and computes $\mathsf{Ct}^* \leftarrow$ **Encrypt**$(\mathsf{Mpk}, (M^*, \rho^*), m^*)$. It returns Ct^* as the challenge ciphertext to $\mathcal{A}$.
4. **Query Phase 2**. $\mathcal{A}$ adaptively issues queries as in **Query Phase 1**, except that it is not allowed to issue an Extract query on input any S satisfying (M^*, ρ^*), nor to issue a Decryption query (Ct^*, S) for any S satisfying (M^*, ρ^*).
5. **Guess**. Finally, $\mathcal{A}$ outputs m'. If $m' = m^*$, $\mathcal{A}$ wins.

$\mathcal{A}$'s advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CAS\text{-}CCA}}(k)$, is its probability of winning the game.

Definition 5 (OW-CAS-CCA Security). *A CP-ABEET scheme is one-way against chosen access structure and chosen ciphertext attacks (*OW-CAS-CCA secure*) if there is no PPT adversary $\mathcal{A}$,* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CAS\text{-}CCA}}(k)$ *is non-negligible.*

If we limit the adversary to submit its challenge access structure (M^*, ρ^*) in an **Init** phase before being given the master public key Mpk, we have the following definition.

Definition 6 (OW-SAS-CCA Security). *A CP-ABEET scheme is one-way against selective access structure and chosen ciphertext attacks (*OW-SAS-CCA secure*) if there is no PPT adversary $\mathcal{A}$,* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}SAS\text{-}CCA}}(k)$ *is non-negligible.*

Indistinguishability against chosen access structure and chosen ciphertext attacks (IND-CAS-CCA): The difference between IND-CAS-CCA and OW-CAS-CCA is that the adversary in the former cannot do equality tests on Ct^*.

1. **Setup**. Same as in OW-CAS-CCA game.
2. **Query Phase 1.** Same as in OW-CAS-CCA game.
3. **Challenge Phase**. $\mathcal{A}$ submits (M^*, ρ^*) as the challenge access structure along with two messages m_0, m_1 of equal length. It is restricted that $\mathcal{A}$ did not ask any Extract query S satisfying (M^*, ρ^*) in **Query Phase 1**. $\mathcal{C}$ then randomly chooses a bit β, and computes $\mathsf{Ct}^* = \mathbf{Encrypt}(\mathsf{Mpk}, (M^*, \rho^*), m_\beta)$. It returns Ct^* to $\mathcal{A}$ as the challenge ciphertext.
4. **Query Phase 2**. Same as in OW-CAS-CCA game, except that $\mathcal{A}$ cannot ask any Trapdoor query S satisfying (M^*, ρ^*) either.
5. **Guess**. Finally, $\mathcal{A}$ outputs a bit β'. If $\beta' = \beta$, $\mathcal{A}$ wins.

$\mathcal{A}$'s advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CAS\text{-}CCA}}(k)$, is the gap between the probability that $\beta' = \beta$ and 1/2, e.g. $|\Pr[\beta' = \beta] - 1/2|$. We have the following definition.

Definition 7 (IND-CAS-CCA Security). *A CP-ABEET scheme is indistinguishable against chosen access structure and chosen ciphertext attacks (*IND-CAS-CCA secure*) if there is no PPT $\mathcal{A}$,* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}CAS\text{-}CCA}}(k)$ *is non-negligible.*

Similarly, if we limit the adversary to submit (M^*, ρ^*) in an **Init** phase before being given the master public key Mpk, we have the following definition.

Definition 8 (IND-SAS-CCA Security). *A CP-ABEET scheme is indistinguishable against selective access structure and chosen ciphertext attacks (*IND-SAS-CCA secure*) if there is no PPT $\mathcal{A}$,* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(k)$ *is non-negligible.*

5 Our CP-ABEET Scheme

5.1 Construction

Our CP-ABEET scheme works as below.

- **Setup**(1^k). The algorithm generates bilinear pairing parameters $(\mathbb{G}, \mathbb{G}_T, p, g, e)$, randomly chooses $\alpha, \alpha', a \in \mathbb{Z}_p$ and calculates $e(g,g)^\alpha, e(g,g)^{\alpha'}, g^a$. It also chooses at random $h_1, \cdots, h_N \in \mathbb{G}$ to represent the N attributes the system supports. Besides, it selects two cryptographic hash functions: $H_1 : \mathbb{G}_T \to \mathbb{G}$ and $H_2 : \mathbb{G}_T \times \mathbb{G}^* \to \{0,1\}^{l_1+l_2}$ where l_1 and l_2 are representation lengths of a $\mathbb{G}$ element and a $\mathbb{Z}_p$ element, respectively. The algorithm outputs a master key pair $(\mathsf{Mpk}, \mathsf{Msk})$, where

$$\mathsf{Mpk} = (g, e(g,g)^\alpha, e(g,g)^{\alpha'}, g^a, h_1, \cdots, h_N),$$

  and $\mathsf{Msk} = (g^\alpha, g^{\alpha'})$.
- **Encrypt**$(\mathsf{Mpk}, (M, \rho), m)$. Suppose that M is an $\ell \times n$ matrix and ρ associates rows of M to attributes. The algorithm chooses a random vector $v = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$ and computes $\lambda_i = v \cdot M_i$ for $i = 1, \cdots, \ell$. Besides, it randomly chooses $u, r_1, \cdots, r_\ell \in \mathbb{Z}_p$, and calculates

$$C = m^u \cdot H_1(e(g,g)^{\alpha s}), \quad C' = g^s, \quad C'' = g^u,$$

$$\forall 1 \leq i \leq \ell,\ C_i = g^{a\lambda_i} \cdot h_{\rho(i)}^{-r_i}, \quad D_i = g^{r_i},$$
$$C^* = (m\|u) \oplus H_2(e(g,g)^{\alpha' s}, C, C', C'', \boldsymbol{E}),$$

where $\boldsymbol{E} = (C_1, D_1, \cdots, C_\ell, D_\ell)$. The algorithm outputs $\mathsf{Ct} = (C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell, C^*)$. We implicitly assume that the access structure is contained in Ct.

- **KeyGen**(Msk, S). The algorithm randomly chooses $t, t' \in \mathbb{Z}_p$, and calculates
$$SK : K = g^{\alpha} \cdot g^{at}, L = g^t, \{K_x = h_x^t\}_{x \in S},$$
$$SK' : K' = g^{\alpha'} \cdot g^{at'}, L' = g^{t'}, \{K'_x = h_x^{t'}\}_{x \in S}.$$
It outputs $\mathsf{Sk}_S = (SK, SK')$.
- **Trapdoor**(Msk, S). The algorithm randomly chooses $t \in \mathbb{Z}_p$, and calculates
$$\mathsf{Td}_S = (K = g^{\alpha} \cdot g^{at}, L = g^t, \{K_x = h_x^t\}_{x \in S}).$$
It outputs Td_S.
- **Test**$(\mathsf{Ct}_A, \mathsf{Td}_{S_A}, \mathsf{Ct}_B, \mathsf{Td}_{S_B})$. Assume that S_A (S_B, resp.) is an authorized set of (M_A, ρ_A) of Ct_A ((M_B, ρ_B) of Ct_B, resp.). Define $I_A = \{i : \rho_A(i) \in S_A\}$ and the set of reconstruction constants $\{w_{A,i} \in \mathbb{Z}_p\}_{i \in I_A}$. The secret s_A can be reconstructed as $s_A = \sum_{i \in I_A} w_{A,i} \cdot \lambda_{A,i}$, where $\{\lambda_{A,i}\}$ are valid shares of s_A w.r.t. M_A. We define I_B and $\{w_{B,i} \in \mathbb{Z}_p\}_{i \in I_B}$ similarly. Parse $\mathsf{Ct}_A = (C_A, C'_A, C''_A, C_{A,1}, D_{A,1}, \cdots, C_{A,\ell}, D_{A,\ell}, C^*_A)$ and $\mathsf{Ct}_B = (C_B, C'_B, C''_B, C_{B,1}, D_{B,1}, \cdots, C_{B,\ell}, D_{B,\ell}, C^*_B)$. The algorithm computes
$$\begin{aligned} X_{sub_A} &= \frac{e(C'_A, K_A)}{\prod_{i \in I_A} (e(C_{A,i}, L_A) e(D_{A,i}, K_{A,\rho(i)}))^{w_{A,i}}} \\ &= \frac{e(g,g)^{\alpha s_A} e(g,g)^{a s_A t_A}}{\prod_{i \in I_A} e(g,g)^{t_A a \lambda_{A,i} w_{A,i}}} = e(g,g)^{\alpha s_A}. \end{aligned}$$
$X_{sub_B} = e(g,g)^{\alpha s_B}$ is computed similarly. It then calculates
$$X_A = \frac{C_A}{H_1(X_{sub_A})} \text{ and } X_B = \frac{C_B}{H_1(X_{sub_B})}.$$
The algorithm outputs 1 if
$$e(C''_A, X_B) = e(C''_B, X_A)$$
holds, and 0 otherwise.
- **Decrypt**$(\mathsf{Ct}, \mathsf{Sk}_S)$. Parse $\mathsf{Ct} = (C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell, C^*)$. The algorithm uses $SK = (K, L, \{K_x\}_{x \in S})$ (the first part of Sk_S) to compute $X_{sub} = e(g,g)^{\alpha s}$ as in **Test** algorithm, and uses $SK' = (K', L', \{K'_x\}_{x \in S})$ (the other part of Sk_S) to compute $X'_{sub} = e(g,g)^{\alpha' s}$ similarly. Then it computes
$$m\|u \leftarrow C^* \oplus H_2\big(X'_{sub}, C, C', C'', \boldsymbol{E}),$$
where $\boldsymbol{E} = (C_1, D_1, \cdots, C_\ell, D_\ell)$. The algorithm outputs m if
$$C'' = g^u \text{ and } C = m^u \cdot H_1(X_{sub})$$
hold, and 0 otherwise.

5.2 Correctness

- **Test**: Let m_A, u_A (resp. m_B, u_B) be the message and random number contained in Ct_A (resp. Ct_B). We have

$$\begin{aligned} e(C''_A, X_B) &= e(g^{u_A}, C_B/H_1(X_{sub_B})) \\ &= e(g^{u_A}, \frac{m_B^{u_B} H_1(e(g,g)^{\alpha s_B})}{H_1(e(g,g)^{\alpha s_B})}) \\ &= e(g^{u_A}, m_B^{u_B}), \\ e(C''_B, X_A) &= e(g^{u_B}, C_A/H_1(X_{sub_A})) \\ &= e(g^{u_B}, \frac{m_A^{u_A} H_1(e(g,g)^{\alpha s_A})}{H_1(e(g,g)^{\alpha s_A})}) \\ &= e(g^{u_B}, m_A^{u_A}). \end{aligned}$$

  If the messages $m_A = m_B$ and the trapdoors Td_A and Td_B are honestly generated, the following equation holds:

$$\begin{aligned} e(C''_A, X_B) &= e(g^{u_A}, m_B^{u_B}) \\ &= e(g^{u_B}, m_A^{u_{1A}}) \\ &= e(C''_B, X_A). \end{aligned}$$

  This completes the correctness analysis of test algorithm.
- **Decryption**: Denote by $\boldsymbol{E} = (C_1, D_1, \cdots, C_\ell, D_\ell)$. We have

$$\begin{aligned} & C^* \oplus H_2(X'_{sub}, C, C', C'', \boldsymbol{E}) \\ =& (m\|u) \oplus H_2(e(g,g)^{\alpha' s}, C, C', C'', \boldsymbol{E}) \\ & \qquad \oplus H_2(e(g,g)^{\alpha' s}, C, C', C'', \boldsymbol{E}) \\ =& m\|u. \end{aligned}$$

  If the following equations

$$C'' = g^u \text{ and } C = m^u \cdot H_1(X_{sub})$$

  hold, the decryption outputs the correct message. This completes the correctness analysis of decryption algorithm.

6 Security Analysis

In this section we analyze the security of our proposed CP-ABEET scheme under the security models given in Sect. 4.2.

Theorem 1. *If decisional q-parallel BDHE assumption holds, our CP-ABEET scheme achieves OW-SAS-CCA security.*

Proof. Suppose that $\mathcal{A}$ is a PPT adversary against the OW-SAS-CCA security of our CP-ABEET scheme. We build an algorithm $\mathcal{B}$ to solve the decisional q-parallel BDHE problem. $\mathcal{B}$ is given a problem instance $(\mathbf{y}, T)$ (please refer to Def. 3 for the definition of $\mathbf{y}$). Define a bit b which is 0 if $T = e(g,g)^{a^{q+1}s}$, and 1 if T is randomly selected from $\mathbb{G}_T$. $\mathcal{B}$ aims to guess the bit b, and works as below.

1. **Init.** $\mathcal{A}$ sends (M^*, ρ^*) to $\mathcal{B}$ as the challenge access structure.
2. **Setup.** $\mathcal{B}$ randomly chooses $\alpha_1, \alpha_2 \in \mathbb{Z}_p$, and sets $e(g,g)^{\alpha} = e(g,g)^{\alpha_1} \cdot e(g, g^{a^q})$ and $e(g,g)^{\alpha'} = e(g,g)^{\alpha_2} \cdot e(g^a, g^{a^q})$. This implicitly sets $\alpha = \alpha_1 + a^q$ and $\alpha' = \alpha_2 + a^{q+1}$. It chooses at random $z_x \in \mathbb{Z}_p$ for each attribute x. Define a set $X^* = \{i : \rho^*(i) = x\}$. $\mathcal{B}$ sets h_x as
$$h_x = g^{z_x} \prod_{i \in X^*} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^n M^*_{i,n}/b_i}.$$
It gives $\mathsf{Mpk} = (g, e(g,g)^{\alpha}, e(g,g)^{\alpha'}, g^a, h_1, \cdots, h_N)$ to $\mathcal{A}$.
3. **Query Phase 1.** *We restrict that the sets of attributes submitted by* $\mathcal{A}$ *in Extract queries would not satisfy* (M^*, ρ^*). $\mathcal{B}$ maintains two hash tables HT_1, HT_2 which are initially empty, and works as below.
   - H_1 Queries. Given an element $Q \in \mathbb{G}_T$, $\mathcal{B}$ traverses the hash table HT_1 to check if there is a tuple (Q, h_1) in the table, and returns h_1 to $\mathcal{A}$ if so; otherwise, $\mathcal{B}$ randomly chooses a value $h_1 \in \mathbb{G}$, stores (Q, h_1) into HT_1, and returns h_1 to $\mathcal{A}$.
   - H_2 Queries. Given $\mathbf{Q} = (Q, C, C', C'', (C_1, D_1), \cdots, (C_\ell, D_\ell))$ as input, $\mathcal{B}$ traverses the hash table HT_1 to check if there is a tuple (Q, h_1) in the table, and returns h_1 to $\mathcal{A}$ if so; Otherwise, if there is not any (Q, h_2) in table HT_2, $\mathcal{B}$ chooses a random $h_2 \in \{0,1\}^{l_1+l_2}$, and stores (Q, h_2) into HT_2. $\mathcal{B}$ returns h_2 to $\mathcal{A}$.
   - Extract Queries. Given an attribute set S, $\mathcal{B}$ computes Sk_S as below. Let $I = \{i : \rho(i) \in S\}$. $\mathcal{B}$ chooses at random $r \in \mathbb{Z}_p$ and finds a vector $w = (w_1 = -1, w_2, \cdots, w_n) \in \mathbb{Z}_p{}^n$ such that $w \cdot M^*_i = 0$ for all $i \in I$. $\mathcal{B}$ firstly calculates $SK' = (K', L', \{K'_x\})$ of Sk_S.$\mathcal{B}$ implicitly sets t' as
   $$t' = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1}$$
   by computing
   $$L' = g^r \prod_{i=1,\cdots,n} (g^{a^{q+1-i}})^{w_i} = g^{t'}.$$
   Then it calculates K' as
   $$\begin{aligned} K' &= g^{\alpha'} g^{at'} = g^{(\alpha_2 + a^{q+1})} g^{at'} \\ &= g^{\alpha_2} \cdot g^{ar} \prod_{i=2,\cdots,n} (g^{a^{q+2-i}})^{w_i}. \end{aligned}$$

Note that the term $g^{-a^{q+1}}$ of component $g^{at'}$ which cannot be simulated, will cancel out with the term $g^{a^{q+1}}$ of $g^{\alpha'}$. For any $x \in S$, if there is no i s.t. $\rho(i) = x$, $\mathcal{B}$ computes

$$K'_x = {h_x}^{t'} = (g^{z_x})^{t'} = (g^{t'})^{z_x} = L'^{z_x}.$$

Otherwise, define $X = \{i : \rho(i) = x\}$. $\mathcal{B}$ computes K'_x as

$$K'_x = L'^{z_x} \prod_{i \in X} \prod_{j=1,\cdots,n} \Big(g^{(a^j/b_i)r} \cdot \prod_{\substack{k=1,\cdots,n \\ k \neq j}} (g^{a^{q+1+j-k}/b_i})^{w_k} \Big)^{M^*_{i,j}}.$$

Note that the terms g^{a^{q+1}/b_i} will all cancel out due to that $w \cdot M^*_i = 0$. Furthremore, $\mathcal{B}$ randomly chooses a new element $t \in \mathbb{Z}_p$ and computes another part $SK = (K, L, \{K_x\})$ of Sk_S as follows.

$$\begin{aligned} K &= g^{\alpha} \cdot g^{at} \\ &= g^{(\alpha_1 + a^q)} \cdot g^{at} \\ &= g^{\alpha_1} g^{a^q} \cdot g^{at}. \end{aligned}$$

$\mathcal{B}$ then computes $L = g^t$ and $K_x = {h_x}^t$ for $\forall x \in S$. Finally $\mathcal{B}$ returns the secret key $\mathsf{Sk}_S = (K, L, \{K_x\}, K', L', \{K'_x\})$.

- Trapdoor Queries. Given an attribute set S, $\mathcal{B}$ randomly chooses t and computes $\mathsf{Td}_S = (K, L, \{K_x\})$ in the same way as above.
- Decryption Queries. Given (Ct, S), $\mathcal{B}$ parses $\mathsf{Ct} = (C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell, C^*)$, and distinguishes the two cases below.
  (a) Case 1: (M^*, ρ^*) is not satisfied by S. $\mathcal{B}$ computes the corresponding private key Sk_S as in dealing with an Extract query, uses Sk_S to decrypt Ct, and returns the output to $\mathcal{A}$.
  (b) Case 2: (M^*, ρ^*) is satisfied by S. In this case, $\mathcal{A}$ could not get Sk_S. $\mathcal{B}$ firstly computes the trapdoor Td_S as above, and computes

$$X_{sub} = \frac{e(C', K)}{\prod_{i \in I} (e(C_i, L) \cdot e(D_i, K_{\rho(i)}))^{w_i}}.$$

Then $\mathcal{B}$ traverses the hash table HT_1 for the tuple (X_{sub}, h_1) and outputs $\perp$ if not found, and traverses HT_2 to see if there exits a tuple $((X'_{sub}, C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell), h_2)$. If there is no such a tuple in HT_2, $\mathcal{B}$ outputs $\perp$. Otherwise, for each tuple found in HT_2, $\mathcal{B}$ computes

$$m \| u = C^* \oplus h_2,$$

and checks whether the following equations hold:

$$C'' = g^u \text{ and } C = m^u \cdot H_1(X_{sub}).$$

If there is a tuple satisfying the following equation, $\mathcal{B}$ outputs the corresponding message m.

4. **Challenge.** $\mathcal{B}$ chooses at random message m^* and $u^* \in \mathbb{Z}_p$, and computes

$$\hat{C} = (m^*)^{u^*} \cdot H_1(T \cdot e(g^s, g^{\alpha_1})),$$
$$\hat{C}' = g^s \text{ and } \hat{C}'' = g^{u^*}.$$

It randomly chooses $y'_2, \cdots, y'_n$ and uses vector $v = (s, sa+y'_2, sa^2+y'_3, \cdots, sa^{n-1}+y'_n)$ to share the secret s. Let $A_i = \{k : k \neq i \wedge \rho^*(k) = \rho^*(i)\}$. $\mathcal{B}$ chooses at random $r'_1, \cdots, r'_\ell \in \mathbb{Z}_p$, and computes

$$\begin{aligned}\hat{C}_i = {} & h^{r'_i}_{\rho^*(i)} \Big(\prod_{j=2,\cdots,n} (g^a)^{M^*_{i,j} y'_j} \Big) \cdot (g^{b_i \cdot s})^{-z_{\rho^*(i)}} \\ & \cdot \Big(\prod_{k \in A_i} \prod_{j=1,\cdots,n} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M^*_{k,j}} \Big), \\ \hat{D}_i = {} & g^{-r'_i} g^{-sb_i},\end{aligned}$$

and

$$\begin{aligned}\hat{C}^* = (m^* \| u^*) \oplus H_2\big(& T \cdot e(g^s, g^{\alpha_2}), \hat{C}, \hat{C}', \hat{C}'', \\ & \hat{C}_1, \hat{D}_1, \cdots, \hat{C}_\ell, \hat{D}_\ell\big).\end{aligned}$$

$\mathcal{B}$ returns $\mathsf{Ct}^* = (\hat{C}, \hat{C}', \hat{C}'', \hat{C}_1, \hat{D}_1, \cdots, \hat{C}_\ell, \hat{D}_\ell, \hat{C}^*)$ to $\mathcal{A}$.
5. **Query Phase 2.** $\mathcal{B}$ answers $\mathcal{A}$'s queries as in **Query Phase 1**, except that now we restrict $\mathcal{A}$ from issuing any Extract query S satisfying (M^*, ρ^*) and any decryption query (S, Ct^*) with S satisfying (M^*, ρ^*).
6. **Guess.** $\mathcal{A}$ outputs a message m'. $\mathcal{B}$ outputs $b' = 0$ if $m' = m^*$, and $b' = 1$ otherwise.

Below we analyze the probability that $\mathcal{B}$ successfully guesses b. If $T = e(g,g)^{a^{q+1}s}$, the simulation provided by $\mathcal{B}$ is perfect, and the view of $\mathcal{A}$ is the same as that of a real attack. It holds that $\Pr[b' = 0 | b = 0] = \mathrm{Adv}^{\mathsf{OW\text{-}SAS\text{-}CCA}}_{\mathcal{A}}(k)$. If T is randomly selected from $\mathbb{G}_T$, the challenge ciphertext hides m^* perfectly, and the probability that $\mathcal{A}$ outputs the correct message is thus negligible, e.g. $\Pr[b' = 0 | b = 1] = \mathrm{negl}(k)$. Therefore, we have the followings.

$$\begin{aligned}\Pr[b' = b] &= \Pr[b' = 0 \wedge b = 0] + \Pr[b' = 1 \wedge b = 1] \\ &= \frac{1}{2}(\Pr[b' = 0 | b = 0] + 1 - \Pr[b' = 0 | b = 1]) \\ &= \frac{1}{2}(\mathrm{Adv}^{\mathsf{OW\text{-}SAS\text{-}CCA}}_{\mathcal{A}}(k) + (1 - \mathrm{negl}(k))) \\ &= \frac{1}{2} + \frac{1}{2}\mathrm{Adv}^{\mathsf{OW\text{-}SAS\text{-}CCA}}_{\mathcal{A}}(k) - \frac{1}{2}\mathrm{negl}(k).\end{aligned}$$

If $\mathrm{Adv}^{\mathsf{OW\text{-}SAS\text{-}CCA}}_{\mathcal{A}}(k)$ is non-negligible, the probability that $\mathcal{B}$ solves the decisional q-parallel BDHE problem is thus non-negligibly larger than $1/2$, which contradicts the underlying assumption.

Theorem 2. *If decisional q-parallel BDHE assumption holds, our CP-ABEET scheme achieves IND-SAS-CCA security.*

Proof. Suppose that $\mathcal{A}$ is a PPT adversary against the IND-SAS-CCA security of our CP-ABEET scheme. We build an algorithm $\mathcal{B}$ to solve the decisional q-parallel BDHE problem. $\mathcal{B}$ is given a problem instance $(\mathbf{y}, T)$. Define a bit b which is 0 if $T = e(g,g)^{a^{q+1}s}$, and 1 if T is randomly selected from $\mathbb{G}_T$. $\mathcal{B}$ aims to guess the bit b, and works as below.

1. **Init.** $\mathcal{A}$ sends (M^*, ρ^*) to $\mathcal{B}$ as the challenge access structure.
2. **Setup.** $\mathcal{B}$ randomly chooses $\alpha_1, \alpha_2 \in \mathbb{Z}_p$, and computes $e(g,g)^{\alpha} = e(g,g)^{\alpha_1} \cdot e(g^a, g^{a^q})$ and $e(g,g)^{\alpha'} = e(g,g)^{\alpha_2} e(g^a, g^{a^q})$. This implicitly sets $\alpha = \alpha_1 + a^{q+1}$ and $\alpha' = \alpha_2 + a^{q+1}$. It chooses at random $z_x \in \mathbb{Z}_p$ for each attribute x. Define a set $X^* = \{i : \rho^*(i) = x\}$. $\mathcal{B}$ sets h_x as
$$h_x = g^{z_x} \prod_{i \in X^*} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^n M^*_{i,n}/b_i}.$$
It gives $\mathsf{Mpk} = (g, e(g,g)^{\alpha}, e(g,g)^{\alpha'}, g^a, h_1, \cdots, h_N)$ to $\mathcal{A}$.
3. **Query Phase 1.** $\mathcal{B}$ handles H_1, H_2 and Trapdoor queries as in the proof of Theorem 1. Again, we restrict that *the sets of attribute submitted by $\mathcal{A}$ in Extract and Trapdoor queries would not satisfy* (M^*, ρ^*). $\mathcal{B}$ answers the adversary's Extract and Decryption queries as below.
   - Extract Queries. Given an attribute set S, $\mathcal{B}$ chooses at random $r, v \in \mathbb{Z}_p$ and finds a vector $w = (w_1 = -1, w_2, \cdots, w_n) \in \mathbb{Z}_p{}^n$ such that $w \cdot M^*_i = 0$ for all $i \in I$, where $I = \{i : \rho(i) \in S\}$. It generates $SK' = (K', L', \{K'_x\})$ of Sk_S as in the proof of Theorem 1, and calculates another part $SK = (K, L, \{K_x\})$ of Sk_S as follows. $\mathcal{B}$ implicitly sets the value t as
   $$t = v + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1}$$
   by computing
   $$L = g^v \prod_{i=1,\cdots,n} (g^{a^{q+1-i}})^{w_i} = g^t.$$
   Then it calculates K as
   $$\begin{aligned} K &= g^{\alpha} g^{at} = g^{(\alpha_1 + a^{q+1})} g^{at} \\ &= g^{\alpha_1} \cdot g^{av} \prod_{i=2,\cdots,n} (g^{a^{q+2-i}})^{w_i}. \end{aligned}$$
   Note that the term $g^{-a^{q+1}}$ of component g^{at} will cancel out with the term $g^{a^{q+1}}$ of g^{α}.
   For any $x \in S$, if there is no i s.t. $\rho(i) = x$, $\mathcal{B}$ computes
   $$K_x = h_x{}^t = (g^{z_x})^t = (g^t)^{z_x} = L^{z_x}.$$

Otherwise, $\mathcal{B}$ computes K_x as

$$K_x = L^{z_x} \prod_{i \in X} \prod_{j=1,\cdots,n} \Big(g^{(a^j/b_i)r} \cdot \prod_{\substack{k=1,\cdots,n \\ k \neq j}} (g^{a^{q+1+j-k}/b_i})^{w_k} \Big)^{M^*_{i,j}},$$

where $X = \{i : \rho(i) = x\}$. Note that the terms g^{a^{q+1}/b_i} will all cancel out due to that $w \cdot M^*_i = 0$. Finally, $\mathcal{B}$ returns the secret key $\mathsf{Sk}_S = (K, L, \{K_x\}, K', L', \{K'_x\})$.

- Decryption Queries. Given (Ct, S), parse $\mathsf{Ct} = (C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell, C^*)$. $\mathcal{B}$ distinguishes the following two cases.
  (a) Case 1: (M^*, ρ^*)k is not satisfied by S. $\mathcal{B}$ generates the corresponding private key Sk_S as in dealing with an Extract query, uses Sk_S to decrypt Ct, and returns the output to $\mathcal{A}$.
  (b) Case 2: (M^*, ρ^*)k is satisfied by S. In this case, $\mathcal{A}$ could not get Sk_S nor Td_S. $\mathcal{B}$ traverses hash table HT_2 to check if there exists a tuple $((X'_{sub}, C, C', C'', C_1, D_1, \cdots, C_\ell, D_\ell), h_2)$. If there is no such a tuple in HT_2, $\mathcal{B}$ outputs $\perp$. Otherwise, for each satisfied tuple found in HT_2, $\mathcal{B}$ computes

  $$m \| u = C^* \oplus h_2.$$

  If $C'' = g^u$ holds, $\mathcal{B}$ calculates $h'_1 = C/m^u$, and searches hash table HT_1 for a tuple (X_{sub}, h'_1). If found, $\mathcal{B}$ outputs m. Otherwise, $\mathcal{B}$ outputs $\perp$. Notice that there is a case in which X_{sub} is not the correct one used in the generation of Ct. However, because of the randomness of oracle H_1, the probability that the adversary uses a correct X_{sub} value in the generation of a well-formed ciphertext without querying H_1, is negligible.

4. **Challenge.** $\mathcal{A}$ submits two messages m^*_0, m^*_1 with $|m^*_0| = |m^*_1|$. $\mathcal{B}$ randomly chooses $\beta \in \{0, 1\}$ and $u^* \in \mathbb{Z}_p$, and computes

$$\hat{C} = (m^*_\beta)^{u^*} \cdot H_1(T \cdot e(g^s, g^{\alpha_1})),$$
$$\hat{C}' = g^s \text{ and } \hat{C}'' = g^{u^*}.$$

It randomly chooses $y'_2, \cdots, y'_n$ and uses vector $v = (s, sa+y'_2, sa^2+y'_3, \cdots, sa^{n-1}+y'_n)$ to share the secret s. Denote by $A_i = \{k : k \neq i \wedge \rho(k) = \rho(i)\}$. $\mathcal{B}$ chooses random values $r'_1, \cdots, r'_\ell$, and computes

$$\hat{C}_i = h^{r'_i}_{\rho(i)} \Big(\prod_{j=2,\cdots,n} (g^a)^{M^*_{i,j} y'_j} \Big) \cdot (g^{b_i \cdot s})^{-z_{\rho(i)}} \cdot \Big(\prod_{k \in A_i} \prod_{j=1,\cdots,n} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M^*_{k,j}} \Big),$$

$$\hat{D}_i = g^{-r'_i} g^{-sb_i}$$

and

$$\hat{C}^* = (m^*\|u^*) \oplus H_2(T \cdot e(g^s, g^{\alpha_2}), \hat{C}, \hat{C}', \hat{C}'',$$
$$\hat{C}_1, \hat{D}_1, \cdots, \hat{C}_\ell, \hat{D}_\ell).$$

$\mathcal{B}$ returns $\mathsf{Ct}^* = (\hat{C}, \hat{C}', \hat{C}'', \hat{C}_1, \hat{D}_1, \cdots, \hat{C}_\ell, \hat{D}_\ell, \hat{C}^*)$ to $\mathcal{A}$.

5. **Query Phase 2.** $\mathcal{B}$ answers $\mathcal{A}$'s queries as in **Query Phase 1**. Now we restrict $\mathcal{A}$ from issuing any Extract query and Trapdoor query S satisfying (M^*, ρ^*), and any decryption query (S, Ct^*) with S satisfying (M^*, ρ^*).
6. **Guess.** $\mathcal{A}$ outputs a bit β'. $\mathcal{B}$ outputs $b' = 0$ if $\beta' = \beta$, and $b' = 1$ otherwise.

Below we analyze the probability that $\mathcal{B}$ successfully guesses b. In case $T = e(g, g)^{a^{q+1}s}$, the simulation provided by $\mathcal{B}$ is perfect, and the view of $\mathcal{A}$ is the same as that of a real attack. It holds that $\Pr[b' = 0|b = 0] = \frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(k)$. In case T is randomly selected from $\mathbb{G}_T$, Ct^* hides the bit β perfectly, and the probability that $\mathcal{A}$ outputs the correct β is $1/2$. Thus, $\mathcal{B}$ successfully guesses b with probability $1/2$, e.g. $\Pr[b' = 0|b = 1] = 1/2$. Therefore, it holds that

$$\begin{aligned}\Pr[b' = b] &= \Pr[b' = 0 \wedge b = 0] + \Pr[b' = 1 \wedge b = 1] \\ &= \frac{1}{2}(\Pr[b' = 0|b = 0] + 1 - \Pr[b' = 0|b = 1]) \\ &= \frac{1}{2}\left(\frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(k) + \frac{1}{2}\right) \\ &= \frac{1}{2} + \frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(k).\end{aligned}$$

If $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(k)$ is non-negligible, the probability that $\mathcal{B}$ solves the decisional q-parallel BDHE problem is thus non-negligibly larger than $1/2$, which contradicts the decisional q-parallel BDHE assumption.

7 Comparison

We make a detailed comparison of our CP-ABEET with some related schemes in Table 1 (page 17), in terms of running costs, functional properties, assumptions, security level and etc. In the comparison we mainly consider the dominant computation in **Extract**, **Encrypt**, **Decrypt**, and **Test** algorithms, e.g. bilinear pairing evaluation and exponentiation operation. The second to the fifth rows of Table 1 show the computational costs of **Extract**, **Encrypt**, **Decrypt**, and **Test** algorithms. The sizes of a ciphertext and a secret key in the schemes are showed in the sixth and seventh rows. The eighth row indicates whether the scheme is attribute based. The ninth row shows the authorization type of each scheme. The last two rows indicate the underlying assumptions and security levels of the schemes.

As ABEET realizes flexible authorization on equality test over encrypted data, it is more suitable for practical use in secure data management of cloud computing. Therefore, the following comparison focuses on ABEET schemes. From

Table 1. Performance Comparison with Related Schemes

	PKEET [19]	PKEET-FA [10](Type-1)	IBEET [9]	KP-ABEwET [20]	CP-ABE-ET [15]	Our CP-ABEET
Extract	$1E$	$3E$	$2E$	$2A_uE$	$(4+6A_u+12A_u^2)E$	$(4+2A_u)E$
Encrypt	$3E$	$6E$	$6E+2P$	$(2A_u+3)E$	$(2N_U+11)E$	$(3\ell+5)E+2P$
Decrypt	$3E$	$5E$	$2E+2P$	$(2A_u+2)E+2A_uP$	$(8A_u+6)E+12P$	$(2A_u+2)E+(4A_u+2)P$
Test	$2P$	$2E+2P$	$4P$	$2A_uE+2A_uP$	$(8A_u+4)E+14P$	$A_uE+(2A_u+3)P$
Ct_{size}	$2\lvert\mathbb{G}\rvert+1$	$2\lvert\mathbb{G}\rvert+2$	$4\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$	$(4+2A_u)\lvert\mathbb{G}\rvert+2\lvert\mathbb{Z}_p\rvert$	$8\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$	$(2+2\ell)\lvert\mathbb{G}\rvert+2$
Sk_{size}	$1\lvert\mathbb{Z}_p\rvert$	$3\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{G}\rvert$	$2A_u\lvert\mathbb{G}\rvert$	$(4+6A_u)\lvert\mathbb{G}\rvert$	$(4+2A_u)\lvert\mathbb{G}\rvert$
Attibute-Based	No	No	No	Yes	Yes	Yes
Authorization	None	Four Types	Single	Flexible	Flexible	Flexible
Assumption	BDH	BDH	BDH	tDBDH	DLIN	q-parallel BDHE
Security	IND-CCA	IND-CCA	OW-ID-CCA	OW-CCA & T-CCA	IND-ID-CPA	OW-SAS-CCA &IND-SAS-CCA

1. T-CCA security of KP-ABEwET means *testability against chosen-ciphertext attack of authorization under the chosen sets of attributes [20]*.
2. In Wang's scheme [15], N_U is the amount of attributes in their system.
3. We denote by A_u the number of attributes used in **Extract**, **Encrypt**, **Decrypt** and **Test** algorithms, and use $|\mathbb{G}|$ and $|\mathbb{Z}_p|$ to denote the length of element representation in $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. In our CP-ABEET scheme, ℓ denotes the number of rows in M.
4. Both the IND-ID-CPA model in [15] and the OW-SAS-CCA and IND-SAS-CCA models in our scheme consider the selective access structure, in which the attacker submits its challenge access structure before seeing public parameters.

Table 1 we know that our CP-ABEET scheme provides the best security guarantee among all the three attribute-based encryption schemes supporting equality test. The computational complexity of **Encrypt** algorithm in our scheme is related to ℓ, number of rows in M. The computational costs of **Extract**, **Decrypt** and **Test** algorithms in our scheme are related to the number A_u of attributes, which has the same order of magnitude of the number I_n of elements in set I defined in **Decrypt** algorithm. Besides, the storage costs of Ct and Sk are related to ℓ and A_u. In Wang et al.'s scheme [15], A_u is related to L_1, which is the number of wildcards defined in their scheme. The two numbers have the same order of magnitude. Besides, our CP-ABEET scheme enjoys OW-SAS-CCA AND IND-SAS-CCA security properties, which are of CCA-type. While Wang et al.'s scheme only achieves IND-ID-CPA security, which is of CPA-type. Notice that, both of the two schemes are provably secure in selective access structure model, in which the adversary is required to submit its challenge access structure before seeing the public parameters. Therefore, our scheme has better security than Wang et al.'s scheme.

Table 1 shows that our scheme has efficiency similar to Zhu et al.'s KP-ABEwET [20]. Compared with CP-ABE-ET [15], our CP-ABEET is more secure and has a more concise construction. Specifically, the key generation in our scheme is more efficient than CP-ABE-ET [15], and the attribute-related secret key is much shorter as well. As for the computational efficiency of **Encrypt**, **Decrypt** and **Test** algorithms, our scheme is comparable with CP-ABE-ET [15] when the number of attributes used in these algorithms is not large. Algorithms **Decrypt** and **Test** would be less efficient when the access policy is complex or the number of attributes is large, which is the price of security improvement of our scheme.

8 Conclusion

We presented a novel construction of CP-ABEET, which combines advantages of both CP-ABE and PKEET. It supports flexible authorization and can be used to implement label classification for encrypted data efficiently. Our construction is more secure than the only known CP-ABEET scheme. Besides, it is more efficient in key generation cost and secret key size. We proved that our CP-ABEET satisfies one-wayness security if adversary has trapdoors, and indistinguishability if adversary is not given trapdoors. Our CP-ABEET scheme is a suitable solution to encrypted data classification with flexible access control.

References

1. N. Attrapadung, B. Libert, and E. D. Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography*, pages 90–108, Taormina, March 2011. Springer.
2. A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* Technion-Israel Institute of technology, Faculty of computer science, Israel, 1996.
3. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, Oakland, May 2007. IEEE Computer Society.
4. V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute based encryption. In *International Colloquium on Automata, Languages, and Programming*, pages 579–591, Reykjavik, July 2008. Springer.
5. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, Alexandria, October 2006. ACM.
6. M. Green, S. Hohenberger, B. Waters, et al. Outsourcing the decryption of ABE ciphertexts. In *Proceedings of USENIX Security Symposium*, pages 34–34, San Francisco, August 2011. USENIX Association.
7. J. Han, W. Susilo, Y. Mu, and J. Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, 23(11):2150–2162, Nov 2012.
8. A. Kapadia, P. P. Tsang, and S. W. Smith. Attribute-based publishing with hidden credentials and hidden policies. In *NDSS*, volume 7, pages 179–192, San Diego, February 2007. The Internet Society.
9. S. Ma. Identity-based encryption with outsourced equality test in cloud computing. *Information Sciences An International Journal*, 328(C):389–402, 2016.
10. S. Ma, Q. Huang, M. Zhang, and B. Yang. Efficient public key encryption with equality test supporting flexible authorization. *IEEE Transactions on Information Forensics and Security*, 10(3):458–470, 2015.
11. S. Ma, M. Zhang, Q. Huang, and B. Yang. Public key encryption with delegated equality test in a multi-user setting. *The Computer Journal*, 58(4):986–1002, 2015.

12. T. Nishide, K. Yoneyama, and K. Ohta. ABE with partially hidden encryptor-specified access structure. acns08, lncs 5037, 2008.
13. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *International Conference on Theory and Applications of Cryptographic Techniques*, pages 457–473, Aarhus, May 2005. Springer.
14. Q. Tang. Towards public key encryption scheme supporting equality test with fine-grained authorization. In *Australasian Conference on Information Security and Privacy*, pages 389–406, Melbourne, July 2011. Springer, Springer.
15. Q. Wang, L. Peng, H. Xiong, J. Sun, and Z. Qin. Ciphertext-policy attribute-based encryption with delegated equality test in cloud computing. *IEEE Access*, 6:760–771, 2018.
16. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 53–70, Taormina, March 2011. Springer.
17. L. Wu, Y. Zhang, K. K. R. Choo, and D. He. Efficient identity-based encryption scheme with equality test in smart city. *IEEE Transactions on Sustainable Computing*, 3(1):44–55, 2018.
18. T. Wu, S. Ma, Y. Mu, and S. Zeng. Id-based encryption with equality test against insider attack. In *Australasian Conference on Information Security and Privacy*, pages 168–183, Auckland, July 2017. Springer.
19. G. Yang, C. H. Tan, Q. Huang, and D. S. Wong. Probabilistic public key encryption with equality test. In *International Conference on Topics in Cryptology*, volume 5985, pages 119–131, San Francisco, March 2010. Springer.
20. H. Zhu, L. Wang, H. Ahmad, and X. Niu. Key-policy attribute-based encryption with equality test in cloud computing. *IEEE Access*, 5:20428–20439, 2017.