# Attribute-Based Equality Test over Encrypted Data without Random Oracles

Yuanhao Wang

Yuzhao Cui

Qiong Huang

Hongbo Li

Jianye Huang
*University of Wollongong*, jh207@uowmail.edu.au

*See next page for additional authors*

# Attribute-Based Equality Test over Encrypted Data without Random Oracles

## Abstract

© 2013 IEEE. Sensitive data would be encrypted before uploading to the cloud due to the privacy issue. However, how to compare the encrypted data efficiently becomes a problem. Public Key Encryption with Equality Test (PKEET) provides an efficient way to check whether two ciphertexts (of possibly different users) contain the same message without decryption. As an enhanced variant, Attribute-based Encryption with Equality Test (ABEET) provides a flexible mechanism of authorization on the equality test. Most of the existing ABEET schemes are only proved to be secure in the random oracle model. Their security, however, would not be guaranteed if random oracles are replaced with real-life hash functions. In this work, we propose a construction of CP-ABEET scheme and prove its security based on some reasonable assumptions in the standard model. We then show how to modify the scheme to outsource complex computations in decryption and equality test to a third-party server in order to support thin clients.

## Disciplines

Engineering | Science and Technology Studies

## Publication Details

Wang, Y., Cui, Y., Huang, Q., Li, H., Huang, J. & Yang, G. (2020). Attribute-Based Equality Test over Encrypted Data without Random Oracles. IEEE Access, 8 32891-32903.

## Authors

Yuanhao Wang, Yuzhao Cui, Qiong Huang, Hongbo Li, Jianye Huang, and Guomin Yang

Received January 14, 2020, accepted January 30, 2020, date of publication February 12, 2020, date of current version February 25, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2973459*

# Attribute-Based Equality Test Over Encrypted Data Without Random Oracles

**YUANHAO WANG[1], YUZHAO CUI[1], QIONG HUANG[1,2], (Member, IEEE), HONGBO LI[1], JIANYE HUANG[3], AND GUOMIN YANG[3], (Senior Member, IEEE)**
[1]College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[2]Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou 510000, China
[3]School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia

Corresponding author: Qiong Huang (qhuang@scau.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872152, in part by the Major Program of Guangdong Basic and Applied Research under Grant 2019B030302008, and in part by the Guangdong Program for Special Support of Top-Notch Young Professionals under Grant 2015TQ01X796.

**ABSTRACT** Sensitive data would be encrypted before uploading to the cloud due to the privacy issue. However, how to compare the encrypted data efficiently becomes a problem. Public Key Encryption with Equality Test (PKEET) provides an efficient way to check whether two ciphertexts (of possibly different users) contain the same message without decryption. As an enhanced variant, Attribute-based Encryption with Equality Test (ABEET) provides a flexible mechanism of authorization on the equality test. Most of the existing ABEET schemes are only proved to be secure in the random oracle model. Their security, however, would not be guaranteed if random oracles are replaced with real-life hash functions. In this work, we propose a construction of CP-ABEET scheme and prove its security based on some reasonable assumptions in the standard model. We then show how to modify the scheme to outsource complex computations in decryption and equality test to a third-party server in order to support thin clients.

**INDEX TERMS** Attribute-based encryption, equality test, outsourced decryption, standard model, deduplication.

## I. INTRODUCTION

The rapid development of cloud computing has brought a variety of convenient services to enterprises and individuals, including cloud storage. Users can upload massive data to the cloud, saving storage overhead while effectively avoiding data loss. Considering the privacy of the data, users generally prefer to encrypt private data and store it in the cloud instead of storing it directly in plaintext form. This also makes it inconvenient for users to search for the data they want in the traditional method. An easy way to address it is to download the files locally, decrypt them, and then search over them. However, it is not practical because it requires a large computation and storage cost. In order to solve the above problems, searchable encryption [1], [2] emerged.

As time goes by, the more files the users upload, the greater possibility of data redundancy is in the cloud, i.e. the encrypted version of the data uploaded by the user may be the same. This kind of data redundancy will bring a great storage burden to cloud computing. Therefore, it is necessary to find and delete duplicated files to optimize the cloud storage. Encrypted data deduplication has attracted many researchers' attention. The technique of **checking whether two ciphertexts contain the same message** is a key to this problem. In addition, new data management requirements arise when considering enterprise data storage. Access control for (encrypted) data also needs to be considered in the enterprise. In a large company, access control of (encrypted) data can be staggered. It is necessary that users with different responsibilities (that is, attributes) have access to the corresponding encrypted data.

*Public Key Encryption with Equality Test* (PKEET), introduced by Yang *et al.* [3], is a variant of *Public Key Encryption with Keyword Search* (PEKS) [1]. It allows the server to check whether two ciphertexts generated under (possibly) different public keys contain the same message without decryption, which is not supported by PEKS. However, [3] allows anyone to execute the equality test, which runs the risk of privacy leakage. To solve this issue, Tang [4]–[6] and

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

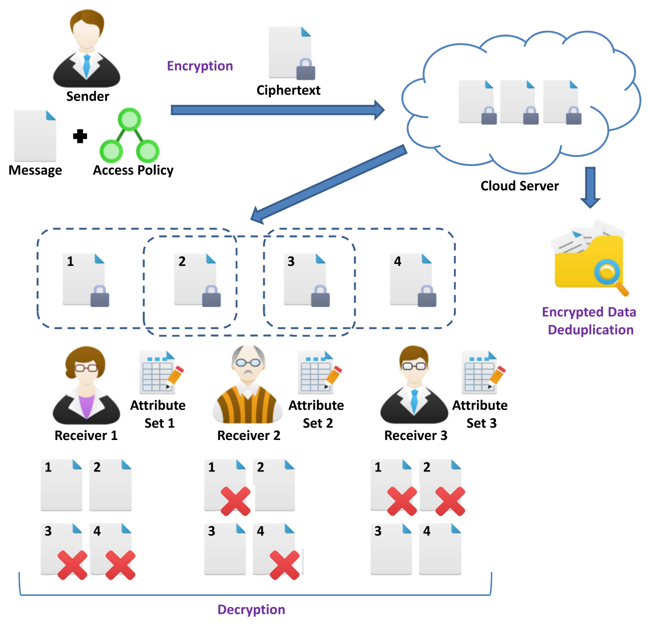


**FIGURE 1.** System model of CP-ABEET.

Ma *et al.* [7] designed different PKEET schemes supporting different kinds of authorization mechanisms.

It is well known that *Attribute-based Encryption* (ABE) enjoys the advantage of flexible access control. The combination of ABE and PKEET simplifies the key management of PKEET and makes its authorization more flexible. Recently, Zhu *et al.* [8] introduced the notion of *Key-policy Attribute-based Encryption with Equality Test* (KP-ABEET). Wang *et al.* [9] and Cui *et al.* [10] studied the ciphertext-policy counterpart and presented their constructions of *Ciphertext-policy Attribute-based Encryption with Equality Test* (CP-ABEET). Take CP-ABEET as an example, it embeds an access policy in the encryption of a message, so that only the authorized receiver whose attribute set satisfies the embedded policy could successfully decrypt and test the ciphertexts.

CP-ABEET can effectively solve the aforementioned problems. Figure 1 illustrates the system model of CP-ABEET. The company sets different attributes (such as financial data, warehouse data), and assigns the private key to each employee according to their responsibilities in form of a set of attributes. Data is encrypted with an access policy embedded. When the attribute set satisfies the access policy, the employee can decrypt the data and process it. (For example, if the attributes of Receiver 2 only match files 2 and 3, he can only decrypt files 2 and 3, but not files 1 and 4.) This also means that there is no need to re-encrypt the data if the attributes of employees change as a result of a job change. New employees can also directly process the data which has been encrypted before they entered the company. The third-party server periodically checks the encrypted data, deletes duplicate data and frees up storage space. In this process, the server cannot extract the information contained in the encrypted data, and encrypted data deduplication does not affect the use of data.

## A. RELATED WORKS

The notion of PKEET was introduced by Yang *et al.* [3] in 2010 as a new variant of searchable encryption mechanism. A fascinating feature of PKEET is that users could check whether two ciphertexts contain the same message *without decryption*. In [3], any entity can perform the equality test on ciphertexts. Due to the lack of access control, there is a risk of information leakage on users' private data. Therefore, Tang [4] proposed the notion of *Fine-grained authorization policy PKEET* (FG-PKEET) to realize the accurate authorization, which only allows two authorized users to perform the equality test. Furthermore, Ma *et al.* [11] proposed *Public key encryption with delegated equality test* (PKE-DET), which only allows the delegated party to test. To make the authorization more flexible, Ma *et al.* [7] proposed a flexible PKEET scheme, which supports four types of authorization. Subsequently, a variety of enhanced schemes [12]–[14] have been proposed to improve security. Zhang *et al.* [12] proposed an efficient PKEET scheme under a specific cryptographic assumption in the standard model.

To solve the problem of complex certificate management in the PKI setting, Ma [15] combined *Identity-based Encryption* (IBE) with PKEET and introduced the notion of *Identity-based Encryption with Equality Test* (IBEET). Users in IBEET scheme use their identity-related keys to generation the trapdoor, which thereby achieves the equality test on its ciphertexts. However, if the server is *curious*, it may illegally benefit from launching a brute force attack against the encrypted data, because ciphertexts can be generated publicly. To solve this problem, Wu *et al.* [16] presented an IBEET scheme against insider attacks. Later, Wu *et al.* [17] proposed another efficient IBEET scheme which reduces the use of time-consuming Hash-to-Point function. In their scheme, it is restricted that only particular keywords can be tested in order to improve the security level.

As an extension of IBE, ABE [18] supports a more flexible authorization mechanism. There are two variants of ABE: Key-policy Attribute-based Encryption (KP-ABE) [19]–[21] and Ciphertext-policy Attribute-based Encryption (CP-ABE) [22]–[24]. In the former, each user is associated with an access policy, and encryption is done w.r.t. an attribute set; in the latter, each user is associated with a set of attributes, and encryption is done w.r.t. an access policy. In each variant, only if the attribute set satisfies the access policy will the decryption succeed. However, ABE schemes suffers from the problem of complex computation. Complexity of ABE schemes usually increases along with the access policy. Green *et al.* [25] suggested securely outsourcing the heavy computation in decryption of an ABE ciphertext to a third-party server, and proposed a concrete scheme, which significantly reduces the overhead of users.

Zhu *et al.* [8] first proposed the construction of KP-ABEET scheme, which is a combination of KP-ABE and PKEET, which provides a more flexible authorization mechanism than previous works. Later, Wang *et al.* [9] proposed a construction of CP-ABEET scheme. Recently, Cui *et al.* [10] proposed another CP-ABEET scheme, which enhances the security of [9]. Then, Cui *et al.* [26] provided another CP-ABEET scheme, which supports to outsource the dominating computations of decryption and equality test to a third-party. Its security is proved in the random oracle model. However, real-life hash function is a deterministic algorithm, which cannot guarantee that the output of the algorithm is completely random and uniformly distributed. If we replace the random oracles with real-life hash functions, the security may no longer be guaranteed. *How to construct a secure and efficient ABEET scheme in standard model* remains an open problem.

### B. OUR CONTRIBUTIONS

In this paper, we study the construction of CP-ABEET in the standard model.

- We propose a new CP-ABEET scheme, which is inspired by Zhang *et al.* [12] and adopts the technique of Lai *et al.* [27] in constructing CCA-secure PKE scheme to eliminate the rely on the random oracle heuristic. Specifically, to encrypt a message, we use a linear secret sharing scheme to share a secret random value $s$, and use $s$ to hide both the message and its hash. Then we use Lai et al.'s technique to ensure the ciphertext's integrity. In both the decryption and the test algorithm, one should first use the decryption key or the trapdoor to reconstruct (an exponentiated form of) the random $s$, and then recover the message or its hash value.
- We prove the security of our CP-ABEET scheme in the standard model based on some reasonable mathematical assumptions. Namely, an unauthorized adversary could not distinguish which message is encrypted for a given ciphertext, while an authorized adversary should not be able to recover the message from a given ciphertext.
- In order to support thin clients (and resource-limited devices), we modify the scheme to outsource complex computations in decryption and equality test to a third-party server, and present an outsourced CP-ABEET scheme.
- We implement our schemes using Java Pairing-Based Cryptography (JPBC) library. Experiment results show that they have comparable and even better efficiency than their counterparts in the random oracle model.

### C. PAPER ORGANIZATION

We introduce some necessary preliminaries in Sect. II, and give the definition of CP-ABEET scheme and its security models in Sect. III. We describe our concrete construction of CP-ABEET scheme in Sect. IV, and prove its security in Sect. V. The outsourced construction of CP-ABEET scheme is given in Sect. VI. We provide a comparison of our schemes with some typical related schemes in the literature in Sect. VII. Experiment results are also given here. Finally, we conclude the paper in Sect. VIII.

## II. PRELIMINARIES

### A. ACCESS STRUCTURE

*Definition 1 (Access Structure [28]):* 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*.

In this paper, we consider monotone access structures. We use attributes to represent parties, and represent the authorized set of parties in access structure $\mathbb{A}$ sets of attributes.

*Definition 2 (Linear Secret Sharing Scheme [29]):* We say a secret sharing scheme $\Pi$ over a set of parties $\mathbb{P}$ is *linear* (over $\mathbb{Z}_p$) if the following 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 $\Pi$, and $(Mv)_i$ is the share belonging to party $\rho(i)$.

There is an efficient *linear reconstruction* 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 $\Pi$ [29].

Same as [28], 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 $\forall i \in I, w \cdot M_i = 0$.

### B. 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) $e(g, g) \neq 1_{\mathbb{G}_T}$; and (3) $\forall g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2)$ can be computed in polynomial time.

### C. MATHEMATICAL ASSUMPTION

Decisional $q$-parallel Bilinear Diffie-Hellman Exponent (Decisional $q$-BDHE) assumption [28] is defined as follows. Suppose $\mathbb{G}$ is a group of prime order $p$, and $g$ is a generator. Randomly choose $s, a, b_1, \cdots, b_q \in \mathbb{Z}_p$. If an adversary is given

$$\mathbf{y} := \Big(\mathbb{G}, p, g, g^s, g^a, \cdots, g^{(a^q)}, , g^{(a^{q+2})}, \cdots, g^{(a^{2q})},$$
$$\forall_{1 \le j \le q}, g^{s \cdot b_j}, g^{a/b_j}, \cdots, g^{(a^q/b_j)}, g^{(a^{q+2}/b_j)}, \cdots, g^{(a^{2q}/b_j)},$$
$$\forall_{1 \le j,k \le q, k \neq j}, g^{(a \cdot s \cdot b_k/b_j)}, \cdots, g^{(a^q \cdot s \cdot b_k/b_j)}\Big),$$

it could not distinguish $e(g, g)^{a^{q+1}s}$ from a random element $R \in \mathbb{G}_T$.

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

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

## III. CIPHERTEXT-POLICY ATTRIBUTE-BASED ENCRYPTION WITH EQUALITY TEST

### A. DEFINITION

*Definition 4 (CP-ABEET):* A CP-ABEET scheme is defined by the following PPT algorithms.

- $\mathsf{Setup}(1^k, U)$: It takes as input a security parameter $1^k$ and the maximal number $U$ of attributes in the system, and returns the system parameters $\mathcal{SP}$ and a master secret key $\mathsf{Msk}$.
- $\mathsf{KeyGen}(\mathcal{SP}, \mathsf{Msk}, S)$: It takes as input $\mathcal{SP}$, $\mathsf{Msk}$ and a set $S$ of attributes, and returns a private key $\mathsf{Sk}_S$.
- $\mathsf{Enc}(\mathcal{SP}, (M, \rho), m)$: It takes as input $\mathcal{SP}$, an access structure $(M, \rho)$ and a message $m$, and returns a ciphertext $\mathsf{Ct}$.
- $\mathsf{Dec}(\mathsf{Ct}, \mathsf{Sk}_S)$: It takes as input a ciphertext $\mathsf{Ct}$ and a private key $\mathsf{Sk}_S$, and returns a plaintext $m$ or a special symbol $\perp$ indicating decryption failure.
- $\mathsf{Trapdoor}(\mathcal{SP}, \mathsf{Msk}, S)$: It takes as input $\mathcal{SP}$, $\mathsf{Msk}$ and a set $S$ of attributes, and returns a trapdoor $\mathsf{Td}_S$.
- $\mathsf{Test}(\mathsf{Ct}_A, \mathsf{Td}_{S_A}, \mathsf{Ct}_B, \mathsf{Td}_{S_B})$: It takes as input a ciphertext $\mathsf{Ct}_A$ and a trapdoor $\mathsf{Td}_{S_A}$ of user $A$, and a ciphertext $\mathsf{Ct}_B$ and a trapdoor $\mathsf{Td}_{S_B}$ of user $B$, and returns 1 if $\mathsf{Ct}_A$ and $\mathsf{Ct}_B$ contain the same plaintext, and 0 otherwise.

### B. SECURITY MODELS

Below we define a security property of CP-ABEET, called *one-wayness against selective access structure and chosen ciphertext attacks* ($\mathsf{OW\text{-}SAS\text{-}CCA}$) security against authorized adversaries. The adversary cannot recover the message from a given ciphertext even if it is given the corresponding trapdoor.

**Game-I**: Let $\mathcal{A}$ be an authorized adversary.

1) **Setup**: $\mathcal{A}$ chooses a challenge access structure $(M^*, \rho^*)$ and submits it to $\mathcal{C}$. Then $\mathcal{C}$ generates $\mathcal{SP}$ and $\mathsf{Msk}$, publishes $\mathcal{SP}$ and keeps $\mathsf{Msk}$ secret.
2) **Query Phase 1**: $\mathcal{A}$ is allowed to issue the following queries for polynomially many times.
    - *Private key Query*: Given an attribute set $S$, it returns the corresponding decryption key $\mathsf{Sk}_S$.
    - *Trapdoor Query*: Given an attribute set $S$, it returns the corresponding trapdoor $\mathsf{Td}_S$.
    - *Decryption Query*: Given an attribute set $S$ and a ciphertext $\mathsf{Ct}$, it returns the corresponding message $m$ or $\perp$ indicating decryption failure.
3) **Challenge**: $\mathcal{C}$ randomly chooses a message $m^*$ from the message space, computes the challenge ciphertext $\mathsf{Ct}^* = \mathsf{Enc}(\mathcal{SP}, (M^*, \rho^*), m^*)$, and returns $\mathsf{Ct}^*$ to $\mathcal{A}$.
4) **Query Phase 2**: Same as **Query Phase 1**.
5) **Guess**: Finally, $\mathcal{A}$ outputs a message $m'$, and wins the game if $m' = m^*$ and $\mathcal{A}$ did not issue an *Private key query* on input any attribute set $S$ satisfying $(M^*, \rho^*)$ nor a *Decryption query* on input $(\mathsf{Ct}^*, S)$ for any attribute set $S$ satisfying $(M^*, \rho^*)$.

The advantage of $\mathcal{A}$ in the game above, denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}SAS\text{-}CCA}}(1^k)$, is defined to be the probability that it wins the game.

*Definition 5 ($\mathsf{OW\text{-}SAS\text{-}CCA}$ Security):* A CP-ABEET scheme is $\mathsf{OW\text{-}SAS\text{-}CCA}$ *secure* if for any PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}SAS\text{-}CCA}}(1^k)$ is negligible.

Next we define another security property of CP-ABEET, called *indistinguishability against selective access structure and chosen ciphertext attacks* ($\mathsf{IND\text{-}SAS\text{-}CCA}$) security against unauthorized adversaries. Without the corresponding trapdoor, the adversary cannot distinguish a given ciphertext is the encryption of which message.

**Game-II**: Let $\mathcal{A}$ be an unauthorized adversary.

1) **Setup**: Same as that in $\mathsf{OW\text{-}SAS\text{-}CCA}$ game.
2) **Query Phase 1**: Same as that in $\mathsf{OW\text{-}SAS\text{-}CCA}$ game.
3) **Challenge**: $\mathcal{A}$ submits two equal-length messages $m_0^*, m_1^*$. $\mathcal{C}$ then randomly chooses a bit $\delta \in \{0, 1\}$, computes the challenge ciphertext $\mathsf{Ct}^* = \mathsf{Enc}(\mathcal{SP}, (M^*, \rho^*), m_\delta^*)$, and returns $\mathsf{Ct}^*$ to $\mathcal{A}$.
4) **Query Phase 2**: Same as that in $\mathsf{OW\text{-}SAS\text{-}CCA}$ game.
5) **Guess**: $\mathcal{A}$ outputs a bit $\delta' \in \{0, 1\}$, and wins the game if $\delta' = \delta$ and $\mathcal{A}$ did not issue any *Private key Query* or *Trapdoor query* on input an attribute set $S$ satisfying $(M^*, \rho^*)$, nor any *Decryption query* on input $(\mathsf{Ct}^*, S)$ for an attribute set $S$ satisfying $(M^*, \rho^*)$.

The advantage of $\mathcal{A}$ in the game above, denoted by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(1^k)$, is defined to the difference between the probability that $\mathcal{A}$ wins the game and $1/2$.

*Definition 6 ($\mathsf{IND\text{-}SAS\text{-}CCA}$ Security):* A CP-ABEET scheme is $\mathsf{IND\text{-}SAS\text{-}CCA}$ *secure* if for any PPT adversary $\mathcal{A}$, its advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}SAS\text{-}CCA}}(1^k)$ is negligible.

## IV. OUR CONCRETE CONSTRUCTION

In this part we present our concrete construction of CP-ABEET scheme. It works as below.

- $\mathsf{Setup}(1^k, U)$: With a security parameter $1^k$ and the maximal number $U$ of attributes in the system, the setup algorithm computes as follows:
    - Choose the groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $p$ along with a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and a generator $g$ of $\mathbb{G}$.
    - Choose the random exponents $a, \alpha, \beta, k_1, k_2, k_3 \in \mathbb{Z}_p$ and $h \in \mathbb{G}$, and compute $g^a$, $g_1 = g^\alpha$, $g_2 = g^\beta$, $u = g^{k_1}, v = g^{k_2}, w = g^{k_3}$.
    - Choose $U$ random group elements $h_1, \cdots, h_U \in \mathbb{G}$ that are associated with the $U$ attributes in the system.
    - Choose two collision-resistant hash functions: $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_T$, $H_2 : \mathbb{G}_T^2 \times \mathbb{G}^{1+2\ell} \rightarrow \mathbb{Z}_p$ where $\ell$ is the number of rows of an LSSS matrix.
    - Publish $\mathcal{SP} = (\mathbb{G}, \mathbb{G}_T, p, e, H_1, H_2, g, g^a, e(g_1, h), e(g_2, h), u, v, w, h_1, \cdots, h_U)$ as the system

parameter. Set $\mathsf{Msk} = (h^{\alpha}, h^{\beta}, k_1, k_2, k_3)$ as the master secret key.

- $\mathsf{KeyGen}(\mathcal{SP}, \mathsf{Msk}, S)$: The key generation algorithm randomly chooses $t, t' \in \mathbb{Z}_p$, computes and returns the following private key $\mathsf{Sk}_S$:

$$\mathsf{Sk}_S = \big(K_d = h^{\alpha} g^{at}, \quad L = g^{t}, \quad \{K_x = h_x^{t}\}_{x\in S},$$
$$K_t = h^{\beta} g^{at'}, \quad L' = g^{t'}, \quad \{K'_x = h_x^{t'}\}_{x\in S}\big).$$

- $\mathsf{Enc}(\mathcal{SP}, (M, \rho), m)$: Let $M$ be an $\ell \times n$ matrix. The encryption algorithm chooses a random vector $\omega = (s, y_2, \cdots, y_n)^{\top} \in \mathbb{Z}_p^n$, and computes $\lambda_i = M_i \cdot \omega$ for $i = 1$ to $\ell$, where $M_i$ is the vector corresponding to the $i$-th row of $M$. Then it randomly chooses $r_0, r_1, \cdots, r_{\ell} \in \mathbb{Z}_p$, and computes

$$C_1 = m \cdot e(g_1, h)^s, \quad C_2 = H_1(m) \cdot e(g_2, h)^s,$$
$$C_3 = g^s, \quad C_4 = r_0, \quad C_5 = (u^T \cdot v^{C_4} \cdot w)^s,$$
$$C_6 = \{(c_i = g^{a\lambda_i} h_{\rho(i)}^{-r_i},\ d_i = g^{r_i})\}_{1\le i\le \ell},$$

where $T = H_2(C_1, C_2, C_3, C_6)$. Then it returns the ciphertext $\mathsf{Ct} = (C_1, C_2, C_3, C_4, C_5, C_6)$, along with a description of $(M, \rho)$.

- $\mathsf{Dec}(\mathsf{Ct}, \mathsf{Sk}_S)$: The decryption algorithm computes $T = H_2(C_1, C_2, C_3, C_6)$, and checks whether $e(C_3, u^T v^{C_4} w) = e(C_5, g)$. If the equation doesn't hold, it outputs $\perp$; otherwise, it computes

$$\begin{aligned}
X &= e(C_3, K_d)/(\prod_{i\in I}(e(c_i, L)e(d_i, K_{\rho(i)}))^{w_i})\\
&= e(g, h)^{\alpha s} e(g, g)^{ast}/(\prod_{i\in I} e(g, g)^{ta\lambda_i w_i})\\
&= e(g, h)^{\alpha s} = e(g_1, h)^s, \text{ and}\\
X' &= e(C_3, K_t)/(\prod_{i\in I}(e(c_i, L')e(d_i, K'_{\rho(i)}))^{w_i})\\
&= e(g, h)^{\beta s} e(g, g)^{ast'}/(\prod_{i\in I} e(g, g)^{t' a\lambda_i w_i})\\
&= e(g, h)^{\beta s} = e(g_2, h)^s.
\end{aligned}$$

Then it computes

$$\hat{H} = \frac{C_2}{X'} \quad \text{and} \quad \hat{m} = \frac{C_1}{X},$$

and outputs $\hat{m}$ if $\hat{H} = H_1(\hat{m})$.

- $\mathsf{Trapdoor}(\mathcal{SP}, \mathsf{Msk}, S)$: The trapdoor algorithm randomly chooses $\hat{t} \in \mathbb{Z}_p$, sets and returns the trapdoor as

$$\mathsf{Td}_S = (\hat{K} = h^{\beta} g^{a\hat{t}},\ \hat{L} = g^{\hat{t}},\ \{\hat{K}_x = h_x^{\hat{t}}\}_{x\in S}).$$

- $\mathsf{Test}(\mathsf{Ct}_A, \mathsf{Td}_{S_A}, \mathsf{Ct}_B, \mathsf{Td}_{S_B})$: W.l.o.g. we assume that the attribute set $S_A$ (resp. $S_B$) associated with $\mathsf{Td}_{S_A}$ (resp. $\mathsf{Td}_{S_B}$) satisfies the access structure $(M_A, \rho_A)$ embedded in ciphertext $\mathsf{Ct}_A$ (resp. $\mathsf{Ct}_B$). Let $I_A \subseteq \{1, 2, \cdots, \ell_A\}$ be defined as $I_A = \{i : \rho_A(i) \in S_A\}$. Then we define the set $\{w_i \in \mathbb{Z}_p\}_{i\in I_A}$ such that if $\{\lambda_i\}$ are valid shares of secret $s$ according to $M_A$, we have $\sum_{i\in I_A} w_i\lambda_i = s$. $I_B$ is defined similarly.

For both $A$ and $B$, the algorithm computes $X'$ as below:

$$\begin{aligned}
X' &= e(C_3, \hat{K})/(\prod_{i\in I}(e(c_i, \hat{L})e(d_i, \hat{K}_{\rho(i)}))^{w_i})\\
&= e(g, h)^{\beta s} e(g, g)^{as\hat{t}}/(\prod_{i\in I} e(g, g)^{\hat{t}a\lambda_i w_i})\\
&= e(g, h)^{\beta s} = e(g_2, h)^s.
\end{aligned}$$

Notice that here we omit the subscripts $A, B$ for simplicity. Then it computes

$$H_A = \frac{C_{2_A}}{X'_A}, \quad H_B = \frac{C_{2_B}}{X'_B},$$

and outputs 1 if $H_A = H_B$, and 0 otherwise.

Correctness of our scheme could be verified straightforward, so we omit it here.

## V. SECURITY ANALYSIS

In this section, we analyze the security of our CP-ABEET scheme and prove in the standard model that our scheme is $\mathsf{OW\text{-}SAS\text{-}CCA}$ secure and $\mathsf{IND\text{-}SAS\text{-}CCA}$ secure under the security models given in Sect. III-B.

### A. OW-SAS-CCA SECURITY

*Theorem 1:* Suppose that the decisional $q$-BDHE assumption holds, our CP-ABEET scheme is $\mathsf{OW\text{-}SAS\text{-}CCA}$ secure against authorized adversaries in standard model.

*Proof:* Based on the security model defined in section III-B, we simulate the security game between the adversary and challenger. Suppose that there exits an adversary $\mathcal{A}$ that attempts to break the $\mathsf{OW\text{-}SAS\text{-}CCA}$ security of our CP-ABEET scheme in standard model. And we define a simulator $\mathcal{B}$ who attempts to solve the decisional $q$-BDHE problem (c.f. Def. 3) from the challenger $\mathcal{C}$. Given a random problem instance $(\mathbf{y}, \mathcal{Z})$, $\mathcal{B}$ aims to decide whether $\mathcal{Z}$ is equal to $e(g, g)^{a^{q+1}s}$ ($b = 0$) or a random element of $\mathbb{G}_T$ ($b = 1$).

This part shows how to build the simulator $\mathcal{B}$.

1) **Setup**: $\mathcal{A}$ chooses a challenge access structure $(M^*, \rho^*)$ and submits it to $\mathcal{B}$. Then $\mathcal{B}$ computes as follows:
   - Choose a group $\mathbb{G}_T$ of prime order $p$ along with a bilinear pairing $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$, and a generator $g$ of $\mathbb{G}$.
   - Choose the elements $\alpha', \beta \in \mathbb{Z}_p$, compute $g_2 = g^{\beta}$, and implicitly set $\alpha = \alpha' + a^q$ by setting
   $$e(g_1, h) = e(g^{\alpha}, h) = e(g^{(a^q)}, h) \cdot e(g, h)^{\alpha'},$$
   where $h = g^a$. Then choose the random elements $x_v, x_w, y_u, y_v, y_w \in \mathbb{Z}_p$ and set
   $$\begin{aligned}
   u &= g^a g^{y_u} = g^{a+y_u},\\
   v &= (g^a)^{x_v} g^{y_v} = g^{ax_v+y_v},\\
   w &= (g^a)^{x_w} g^{y_w} = g^{ax_w+y_w}.
   \end{aligned}$$
   - Choose a random $z_x \in \mathbb{Z}_p$ for each attribute $A_x$ where $1 \le x \le U$. Define $X$ as the set of $i$ where

$\rho^*(i) = x$. Set $h_x$ as

$$h_x = g^{z_x} \prod_{i \in X} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^n M^*_{i,n}/b_i}.$$

Notice that if $X = \emptyset$, we have $h_x = g^{z_x}$.

- Choose two collision-resistant hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_T$ and $H_2 : \mathbb{G}_T^2 \times \mathbb{G}^{1+2\ell} \to \mathbb{Z}_p$, where $\ell$ is the number of rows of an LSSS matrix.
- Publish $\mathcal{SP} = (\mathbb{G}, \mathbb{G}_T, p, e, H_1, H_2, g, g^a, e(g_1, h), e(g_2, h), u, v, w, h_1, \cdots, h_U)$ as the system parameter. Notice that the master secret key $\mathsf{Msk} = (h^\alpha, h^\beta, k_1, k_2, k_3)$ is unknown to $\mathcal{B}$, except for the element $h^\beta$.

2) **Query Phase 1**: $\mathcal{B}$ answers queries from $\mathcal{A}$ as follows. Here we assume that all the queries submitted by $\mathcal{A}$ would not violate the restrictions specified in the game (c.f. Def. 5).

- *Private key query*: Given an attribute set $S$ from $\mathcal{A}$, $\mathcal{B}$ randomly chooses $r, t' \in \mathbb{Z}_p$ and finds a vector $w = (w_1, w_2, \cdots, w_n) \in \mathbb{Z}_p{}^{n^*}$ with $w_1 = -1$ such that $w \cdot M_i = 0$ for all $i \in I$, where $I = \{i : \rho(i) \in S\}$. Such a vector exists according to Def. 2. $\mathcal{B}$ implicitly sets $t$ as

$$t = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1},$$

by defining

$$L = g^r \prod_{i=1,\cdots,n} (g^{a^{q+1-i}})^{w_i} = g^t.$$

Then it computes $K_d$ as

$$K_d = h^\alpha g^{at} = h^{\alpha'} h^{a^q} g^{at} = h^{\alpha'} g^{a^{q+1}} g^{at} = h^{\alpha'} g^{ar} \prod_{i=2,\cdots,n} (g^{a^{q+2-i}})^{w_i}.$$

Next we show how $\mathcal{B}$ computes $\{K_x\}_{x\in S}$. For each $x \in S$, if there is no $i$ such that $\rho(i) = x$, then $\mathcal{B}$ sets

$$K_x = h_x^t = (g^{z_x})^t = (g^t)^{z_x} = L^{z_x};$$

otherwise, there exists one or more mappings between the rows of matrix $M$ and $x \in S$. Let $X$ be the set of $i$ s.t. $\rho(i) = x$. $\mathcal{B}$ sets $K_x$ as

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

Notice that the terms $g^{a^{q+1}/b_i}$ which cannot be simulated would all be cancelled out due to the character that $w \cdot M_i = 0$.

Then $\mathcal{B}$ computes $K_t$, $L'$ and $\{K'_x\}_{x\in S}$ using the method described in Section IV. Finally, $\mathcal{B}$ returns $\mathsf{Sk}_S = \big(K_d, L, \{K_x\}_{x\in S}, K_t, L', \{K'_x\}_{x\in S}\big)$.

- *Trapdoor query*: $\mathcal{B}$ computes and returns $\mathsf{Td}_S$ using the method described in section IV.
- *Decryption query*: Given an attribute set $S$ and a ciphertext $\mathsf{Ct}$, there two cases:

a) Case 1: $S$ does not satisfy $(M^*, \rho^*)$. $\mathcal{B}$ gets $\mathsf{Sk}_S$ from *private key query*, and uses $\mathsf{Sk}_S$ to run the decryption algorithm to decrypt $\mathsf{Ct}$.

b) Case 2: $S$ satisfies $(M^*, \rho^*)$. $\mathcal{B}$ runs $\mathsf{Trapdoor}$ to generate $\mathsf{Td}_S = (K_t, L', \{K'_x\}_{x\in S})$. Parse $\mathsf{Ct}$ as $\mathsf{Ct} = (C_1, C_2, C_3, C_4, C_5, C_6)$. $\mathcal{B}$ computes $T = H_2(C_1, C_2, C_3, C_6)$, and checks whether

$$e(C_3, u^T v^{C_4} w) = e(C_5, g).$$

If the equation does not hold, $\mathcal{B}$ returns $\perp$; otherwise, it continue to check the equation

$$T + C_4\, x_v + x_w = 0.$$

If it holds, $\mathcal{B}$ aborts, and we denote this event by $E_1$; otherwise, $\mathcal{B}$ computes

$$\begin{aligned} X' &= \frac{e(C_3, K_t)}{\prod_{i\in I}(e(c_i, L')e(d_i, K'_{\rho(i)}))^{w_i}} \\ &= \frac{e(g,h)^{\beta s} e(g,g)^{ast'}}{\prod_{i\in I} e(g,g)^{t' a \lambda_i w_i}} \\ &= e(g,h)^{\beta s}, \text{ and} \\ \hat{H} &= \frac{C_2}{X'} = \frac{H_1(m)\cdot e(g_2,h)^s}{e(g,h)^{\beta s}} \\ &= H_1(m). \end{aligned}$$

Then $\mathcal{B}$ chooses $s' \in \mathbb{Z}_p$, and computes

$$P_{1,m} = g_1^{-\frac{Ty_u + C_4\ y_v + y_w}{T + C_4\ x_v + x_w}} (u^T v^{C_4} w)^{s'},$$
$$P_{2,m} = g_1^{-\frac{1}{T + C_4\ x_v + x_w}} g^{s'}.$$

Then it recovers the message by computing

$$\hat{m} = C_1 \cdot \frac{e(C_5, P_{2,m})}{e(C_3, P_{1,m})}.$$

If $\hat{H} = H_1(\hat{m})$, $\mathcal{B}$ returns $\hat{m}$ to $\mathcal{A}$; otherwise, it returns $\perp$.

3) **Challenge**: In this phase, $\mathcal{B}$ randomly chooses a message $m^*$ from the message space, and computes

$$\begin{aligned} C_1^* &= m^* \cdot \mathcal{Z} \cdot e(g^s, h)^{\alpha'}, \\ C_2^* &= H_1(m^*) \cdot e(g^s, h)^{\beta}, \\ C_3^* &= g^s. \end{aligned}$$

Then $\mathcal{B}$ randomly chooses $y'_2, \cdots, y'_n \in \mathbb{Z}_p$ and shares $s$ using the vector $\omega = (s, sa + y'_2, sa^2 + y'_3, \cdots, sa^{n-1} + y'_n)$. Define $A_i$ as the set of all $k \neq i$ but $\rho^*(k) = \rho^*(i)$.

$\mathcal{B}$ chooses $r'_1, \cdots, r'_\ell \in \mathbb{Z}_p$ and generates $C_6^* = \{(c_i,\ d_i)\}_{1\le i\le \ell}$ as follows:

$$c_i = h_{\rho^*(i)}^{r'_i}\left(\prod_{j=2,\cdots,n}(g^a)^{M^*_{i,j}y'_j}\right)\cdot (g^{b_i\cdot s})^{-z_{\rho(i)}}$$
$$\cdot\left(\prod_{k\in A_i}\prod_{j=1,\cdots,n}(g^{a^j\cdot s\cdot(b_i/b_k)})^{M^*_{k,j}}\right),$$
$$d_i = g^{-r'_i}g^{-sb_i}.$$

Then $\mathcal{B}$ computes $T^* = H_2(C_1^*, C_2^*, C_3^*, C_6^*)$ and sets

$$C_4^* = -\frac{T^*+x_w}{x_v},$$
$$C_5^* = (g^s)^{T^*y_u+C_4^*y_v+y_w}.$$

Finally, $\mathcal{B}$ sends $\mathsf{Ct}^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*)$ to $\mathcal{A}$.

4) **Query Phase 2**: In this phase, $\mathcal{B}$ answers queries in the same way as in **Query Phase 1** with the following restriction:
   - Given a decryption query $(\mathsf{Ct}^*, S)$, while $S$ satisfies $(M^*, \rho^*)$, $\mathcal{B}$ returns $\perp$, as $\mathcal{A}$ is not allowed to this query.
   - If $\mathsf{Ct} = (C_1, C_2, C_3^*, C_4^*, C_5^*, C_6^*) \neq \mathsf{Ct}^*$ but $T = H_2(C_1, C_2, C_3^*, C_6^*) = T^*$, we get a collision of hash function $H_2$. In this case $\mathcal{B}$ aborts. We define this event by $E_2$.
   - If $T + C_4\ x_v + x_w = 0$ holds, where $T = H_2(C_1, C_2, C_3, C_6)$, $\mathcal{B}$ aborts. We define this event by $E_3$.

5) **Guess**: Finally, $\mathcal{A}$ outputs a message $m'$. $\mathcal{B}$ outputs $b' = 0$ if $m' = m^*$, indicating that $\mathcal{Z} = e(g,g)^{a^{q+1}s}$, and a random bit $b'$ otherwise.

**Analysis**: In this part, we analyse the events that makes the simulation fail or abort.

- **The failures caused by hash functions**: Firstly, we pay our attention to the one-wayness of $H_1$. In the simulation process, adversary $\mathcal{A}$ has the authorization to query the trapdoor for challenge access structure so that it can obtain the hash value of challenge message $H_1(m^*)$. Adversary $\mathcal{A}$ may learn some information about message $m^*$ from $H_1(m^*)$. In other words, the simulation fails if adversary $\mathcal{A}$ breaks the one-wayness of hash function $H_1$. We define this event as $E_4$ and we have
$$Pr[E_4] \le \varepsilon_{OW},$$
where $\varepsilon_{OW}$ is the probability that adversary $\mathcal{A}$ successfully breaks the one-wayness of $H_1$.
Then we turn to the collision resistance of hash function $H_2$. When $E_2$ occurs during the decryption queries in **Query Phase 2**, there exits a hash collision such that $T = H_2(C_1, C_2, C_3^*, C_6^*) = T^* = H_2(C_1^*, C_2^*, C_3^*, C_6^*)$. We have
$$Pr[E_2] \le \varepsilon_{CR},$$
where $\varepsilon_{CR}$ is the probability that adversary $\mathcal{A}$ successfully breaks the collision resistance of $H_2$.
- **The failures caused by simulation limits**: During the whole simulation process, some events will make it abort in which simulator $\mathcal{B}$ cannot give a logical answer to decryption queries from $\mathcal{A}$.
$E_1$ and $E_3$ occur when the elements of queried ciphertext satisfy the relation: $T + C_4x_v + x_w = 0$. Because that element $T$ depends on the submitted ciphertext and $x_v$, $x_w$ are fixed, the probabilities of $E_1$ and $E_3$ depend on the randomness of $C_4 = r_0$ chosen from $Z_p$. It means that equation $T + C_4x_v + x_w$ happens with probability at most $1/p$ in a single query. We have
$$Pr[E_1 \vee E_3] \le Pr[E_1] + Pr[E_3] \le q_D/p,$$
where $q$ means $\mathcal{A}$ is allowed to issue *Decryption query* for $q$ times.

We obtain the final failure and abortion probability $Pr[F]$ as

$$Pr[F] = Pr[E_1 \vee E_2 \vee E_3 \vee E_4]$$
$$\le \varepsilon_{OW} + \varepsilon_{CR} + q_D/p.$$

Below we analyze the probability that $\mathcal{B}$ successfully guess the value of $b$. If $T = e(g,g)^{a^{q+1}s}$, the simulation provided by $\mathcal{B}$ is perfect, and the in view of $\mathcal{A}$, the challenge ciphertext is the same as a real ciphertext. We have that $\Pr[b' = 0|b = 0] = \mathrm{Adv}_{\mathcal{A}}^{\text{OW-SAS-CCA}}(1^k)\cdot Pr[\neg F]$; otherwise, which means $T$ is a random element of $\mathbb{G}_T$, the challenge ciphertext hides the message perfectly, and the probability that $\mathcal{A}$ outputs the correct message is thus negligible, e.g. $\Pr[b' = 1|b = 1] = 1 - \mathrm{negl}(1^k)\cdot Pr[\neg F]$. Therefore, we have:

$$\begin{aligned}
&Pr[b' = b]\\
&= Pr[b' = 0 \wedge b = 0] + Pr[b' = 1 \wedge b = 1]\\
&= \frac{1}{2}(Pr[b' = 0|b = 0] + Pr[b' = 1|b = 1])\\
&= \frac{1}{2}\big(\mathrm{Adv}_{\mathcal{A}}^{\text{OW-SAS-CCA}}(1^k)\cdot Pr[\neg F] + (1-\mathrm{negl}(1^k)\cdot Pr[\neg F])\big)\\
&= \frac{1}{2} + \big(\frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{\text{OW-SAS-CCA}}(1^k) - \frac{1}{2}\mathrm{negl}(1^k)\big)\cdot Pr[\neg F]\\
&\ge \frac{1}{2} + \big(\frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{\text{OW-SAS-CCA}}(1^k) - \frac{1}{2}\mathrm{negl}(1^k)\big)\\
&\quad\cdot[1 - (\varepsilon_{OW} + \varepsilon_{CR} + q_D/p)],
\end{aligned}$$

where $[1 - (\varepsilon_{OW} + \varepsilon_{CR} + q_D/p)]$ is non-negligible.

If $\mathcal{A}$ breaks the OW-SAS-CCA security of our CP-ABEET scheme with non-negligible advantage, the probability that $\mathcal{B}$ solves the decisional $q$-BDHE problem is thus non-negligibly larger than $\frac{1}{2}$, which contradicts the decisional $q$-BDHE assumption. This completes the proof of Theorem 1. ■

### B. IND-SAS-CCA SECURITY

*Theorem 2:* Suppose that the decisional $q$-BDHE assumption holds, our CP-ABEET scheme is IND-SAS-CCA secure against unauthorized adversaries in standard model.

*Proof:* Based on the security model defined in section III-B, we simulate the security game between the adversary and challenger. Suppose that there exits an adversary $\mathcal{A}$ that attempts to break the IND-SAS-CCA security of our CP-ABEET scheme in standard model. And we define a simulator $\mathcal{B}$ who attempts to solve the decisional $q$-BDHE problem from the challenger $\mathcal{C}$. Given a random problem instance $(\mathbf{y}, \mathcal{Z})$, $\mathcal{B}$ aims to decide whether $\mathcal{Z}$ is equal to $e(g,g)^{a^{q+1}s}$ $(b=0)$ or a random element of $\mathbb{G}_T$ $(b=1)$.

This part shows how to build the simulator $\mathcal{B}$.

1) **Setup**: $\mathcal{A}$ chooses a challenge access structure $(M^*, \rho^*)$ and submits it to $\mathcal{B}$. Then $\mathcal{B}$ generates the system parameters $\mathcal{SP}$ basing on the $q$-BDHE challenge instance $\mathbf{y}$. Firstly, $\mathcal{B}$ randomly chooses elements $\alpha', \beta' \in \mathbb{Z}_p$. Then it implicitly sets $\alpha = \alpha' + a^q$, $\beta = \beta' + a^q$ by setting

$$e(g_1, h) = e(g^\alpha, h) = e(g^{(a^q)}, h) \cdot e(g,h)^{\alpha'},$$
$$e(g_2, h) = e(g^\beta, h) = e(g^{(a^q)}, h) \cdot e(g,h)^{\beta'},$$

where $h = g^a$. Besides, $\mathcal{B}$ chooses random elements $x_v, x_w, y_u, y_v, y_w \in \mathbb{Z}_p$ and sets

$$u = g^a g^{y_u} = g^{a+y_u},$$
$$v = (g^a)^{x_v} g^{y_v} = g^{ax_v+y_v},$$
$$w = (g^a)^{x_w} g^{y_w} = g^{ax_w+y_w}.$$

Then we show how to obtain the group elements $h_1, \cdots, h_U$ by $\mathcal{B}$. Firstly $\mathcal{B}$ chooses a random value $Z_x$ for each $x$ where $1 \le x \le U$. We define $X$ as the set of $i$ where $\rho(i) = x$. Then $\mathcal{B}$ programs $h_x$ as

$$h_x = g^{z_x} \prod_{i \in X} g^{aM_{i,1}/b_i} \cdot g^{a^2 M_{i,2}/b_i} \cdots g^{a^n M_{i,n}/b_i}.$$

Notice that if $X = \emptyset$, $h_x = g^{z_x}$. Then it chooses two cryptographic hash functions: $H_1 : \{0,1\}^* \to \mathbb{G}_T$, $H_2 : \mathbb{G}_T \times \mathbb{G}_T \times \mathbb{G}^{1+2\ell} \to \mathbb{Z}_p$ where $\ell$ is the number of rows in LSSS matrix. Finally, it publishes $\mathcal{SP} = (\mathbb{G}, \mathbb{G}_T, p, e, H_1, H_2, g, g^a, e(g_1, h), e(g_2, h), u, v, w, h_1, \cdots, h_U)$ as the system parameter. Notice that the master secret key $\mathsf{Msk} = (h^\alpha, h^\beta, k_1, k_2, k_3)$ is unknown to $\mathcal{B}$.

2) **Query Phase 1**: In this phase, *Trapdoor query* executes as same as that in the proof of Theorem 1 with another restriction that all the submitted attribute sets cannot satisfy the challenge access structure $(M^*, \rho^*)$.

- *Private key query*: Given an attribute set $S$ from $\mathcal{A}$, $\mathcal{B}$ randomly chooses $r, r' \in \mathbb{Z}_p$ and finds a vector $w = (w_1, w_2, \cdots, w_n) \in \mathbb{Z}_p{}^{n^*}$ with $w_1 = -1$ such that $w \cdot M_i = 0$ for all $i \in I$, where $I = \{i : \rho(i) \in S\}$. Such a vector exists according to Def. 2. $\mathcal{B}$ implicitly sets $t$ as

$$t = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1}$$

by defining

$$L = g^r \prod_{i=1,\cdots,n} (g^{a^{q+1-i}})^{w_i} = g^t.$$

Then it computes $K_d$ as

$$K_d = h^\alpha g^{at} = h^{\alpha'} h^{a^q} g^{at} = h^{\alpha'} g^{a^{q+1}} g^{at}$$
$$= h^{\alpha'} g^{ar} \prod_{i=2,\cdots,n} (g^{a^{q+2-i}})^{w_i}.$$

Next we show how $\mathcal{B}$ computes $\{K_x\}_{x \in S}$. For each $x \in S$, if there is no $i$ such that $\rho(i) = x$, then $\mathcal{B}$ sets

$$K_x = h_x^t = (g^{z_x})^t = (g^t)^{z_x} = L^{z_x};$$

otherwise, there exists one or more mappings between the rows of matrix $M$ and $x \in S$. Let $X$ be the set of $i$ s.t. $\rho(i) = x$. $\mathcal{B}$ sets $K_x$ as

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

Notice that the terms $g^{a^{q+1}/b_i}$ which cannot be simulated would all be cancelled out due to the character that $w \cdot M_i = 0$. To generate the second part of $\mathsf{Sk}_S$, $\mathcal{B}$ implicitly sets the value $t'$ as

$$t' = r' + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q-n+1}$$

by defining

$$L' = g^{r'} \prod_{i=1,\cdots,n} (g^{a^{q+1-i}})^{w_i} = g^{t'}.$$

The elements $K_t$ and $\{K'_x\}_{x \in S}$ could be generated using $r', t'$ in a similar way. Finally, $\mathcal{B}$ returns $\mathsf{Sk}_S$.

- *Decryption query*: In this phase, $\mathcal{B}$ will answer the decryption queries from $\mathcal{A}$. Given an attribute set $S$ and a ciphertext $\mathsf{Ct}$, there two cases:
  a) Case 1: $S$ does not satisfy the challenge access structure $(M^*, \rho^*)$. $\mathcal{B}$ can firstly obtain the corresponding private key $\mathsf{Sk}_S$. Then it uses the $\mathsf{Sk}_S$ to decrypt the queried ciphertext as the $\mathsf{Dec}$ algorithm does.
  b) Case 2: $S$ satisfies the challenge access structure $(M^*, \rho^*)$. $\mathcal{B}$ cannot directly decrypt the queried ciphertext using the corresponding $\mathsf{Sk}_S$. Besides, it has no authorization for the $\mathsf{Td}$. Suppose the submitted ciphertext is $\mathsf{Ct} = (C_1, C_2, C_3, C_4, C_5, C_6)$. First of all, the ciphertext validity should be verified as follows.
  $\mathcal{B}$ computes $T = H_2(C_1, C_2, C_3, C_6)$. Then, it checks whether

  $$e(C_3, u^T v^{C_4} w) = e(C_5, g).$$

  If the equation doesn't hold, the system output $\perp$; otherwise, $\mathcal{B}$ continue to check the following equation:

  $$T + C_4 x_v + x_w = 0.$$

If it holds, $\mathcal{B}$ aborts and we denote this event as $E_1$; otherwise, $\mathcal{B}$ obtains the corresponding $\hat{H}$ and $\hat{m}$ using the similar method described in the proof of OW-SAS-CCA security above. $\mathcal{B}$ chooses a random element $s' \in \mathbb{Z}_p$. Then it computes

$$P_{1,h} = g_2^{-\frac{Ty_u+C_4y_v+y_w}{T+C_4x_v+x_w}}(u^T v^{C_4} w)^{s'},$$
$$P_{2,h} = g_2^{-\frac{1}{T+C_4x_v+x_w}} g^{s'}.$$

Then the message can be recovered as follows

$$\begin{aligned}\hat{H} &= C_2 \cdot \frac{e(C_5, P_{2,h})}{e(C_3, P_{1,h})}\\ &= C_2 \cdot e(g^{as}, (g_2)^{-1})\\ &= H_1(m) \cdot e(g_2, h)^s \cdot e(g^{as}, (g_2)^{-1})\\ &= H_1(m).\end{aligned}$$

If the submitted ciphertext Ct is valid and $m$ is the message encrypted in this ciphertext Ct, $\hat{H}$ is the hash value of $m$. Then $\mathcal{B}$ computes

$$P_{1,m} = g_1^{-\frac{Ty_u+C_4y_v+y_w}{T+C_4x_v+x_w}}(u^T v^{C_4} w)^{s'},$$
$$P_{2,m} = g_1^{-\frac{1}{T+C_4x_v+x_w}} g^{s'}.$$

Then the message can be recovered as follows

$$\begin{aligned}\hat{m} &= C_1 \cdot \frac{e(C_5, P_{2,m})}{e(C_3, P_{1,m})}\\ &= C_1 \cdot e(g^{as}, (g_1)^{-1})\\ &= m \cdot e(g_1, h)^s \cdot e(g^{as}, (g_1)^{-1})\\ &= m.\end{aligned}$$

Correctness of this process can be proven in the same way described in the proof of OW-SAS-CCA security above. If the submitted ciphertext Ct is valid and $m$ is the massage encrypted in Ct, the message can be recovered through this process. If the equation $\hat{H} = H_1(\hat{m})$ holds, $\mathcal{B}$ output $\hat{m}$ to $\mathcal{A}$.

3) **Challenge**: $\mathcal{A}$ randomly chooses two messages $m_0, m_1 \in \mathcal{M}$ and sends them to $\mathcal{B}$. Then $\mathcal{B}$ randomly chooses a bit $\delta \in \{0, 1\}$ and generates the corresponding challenge ciphertext $\mathsf{Ct}^* = \mathsf{Enc}(m_\delta)$ as follows: Firstly, $\mathcal{B}$ computes

$$C_1^* = m_\delta \cdot \mathcal{Z} \cdot e(g^s, h)^{\alpha'},$$
$$C_2^* = H_1(m_\delta) \cdot \mathcal{Z} \cdot e(g^s, h)^{\beta'},$$
$$C_3^* = g^s.$$

Secondly, $\mathcal{B}$ randomly chooses $y_2', \cdots, y_n' \in \mathbb{Z}_p$ and shares the secret using the vector $\omega = (s, sa+y_2', sa^2+y_3', \cdots, sa^{n-1}+y_n')$. Then, $\mathcal{B}$ chooses random values $r_1', \cdots, r_\ell' \in \mathbb{Z}_p$. Besides, for $1 \le i \le n$, we define $A_i$ as the set of all $k \neq i$ where $\rho(i) = \rho(k)$. $\mathcal{B}$ generates $C_6^* = \{(c_i, d_i)\}_{1 \le i \le \ell}$ as follows:

$$c_i = h_{\rho(i)}^{r_i'} \left(\prod_{j=2,\cdots,n} (g^a)^{M^*_{i,j} y_j'}\right) \cdot (g^{b_i \cdot s})^{-z_{\rho(i)}} \cdot \left(\prod_{k \in A_i} \prod_{j=1,\cdots,n} (g^{a^j \cdot s \cdot (b_i/b_k)})^{M^*_{k,j}}\right),$$
$$d_i = g^{-r_i'} g^{-sb_i}.$$

Then, $\mathcal{B}$ computes $T^* = H_2(C_1^*, C_2^*, C_3^*, C_6^*)$ and sets

$$C_4^* = -\frac{T^*+x_w}{x_v},$$
$$C_5^* = (g^s)^{T^*y_u + C_4^* y_v + y_w}.$$

Finally, $\mathcal{B}$ returns $\mathsf{Ct}^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, C_6^*)$ to $\mathcal{A}$.

4) **Query Phase 2**: In this phase, $\mathcal{B}$ answers *Private key query* and *Trapdoor query* in the same way as **Query Phase 1**. And $\mathcal{B}$ answers *decryption query* with the following restriction:
   a) If the submitted ciphertext is equal to the challenge ciphertext $\mathsf{Ct}^*$, $\mathcal{B}$ returns $\perp$.
   b) If $\mathsf{Ct} = (C_1, C_2, C_3^*, C_4^*, C_5^*, C_6^*) \neq \mathsf{Ct}^*$ but $T = H_2(C_1, C_2, C_3^*, C_6^*) = T^*$, which means there exits a hash collision of hash function $H_2$, $\mathcal{B}$ aborts. We define this event as $E_2$.
   c) Otherwise, if the equation $T + C_4 x_v + x_w = 0$ holds where $T$ is as described before, $\mathcal{B}$ aborts. We define this event as $E_3$.

5) **Guess**: $\mathcal{A}$ outputs a guess $\delta' \in \{0, 1\}$. $\mathcal{B}$ outputs $b' = 0$ if $\delta' = \delta$, indicating that $\mathcal{Z} = e(g, g)^{a^{q+1}s}$, and a random bit $b'$ otherwise.

**Analysis**: In this part, we analyse the events that makes the simulation fail or abort.

- **The failures caused by hash functions**: Firstly, we pay our attention to the one-wayness of $H_1$. In the simulation process, $\mathcal{A}$ has no authorization to query the trapdoor for challenge access structure so that it cannot obtain the hash value of challenge message $H_1(m_\delta)$. So adversary successfully breaking one-wayness of $H_1$ won't reveal any information of challenge message.
  Then we turn to the collision resistance of hash function $H_2$. When $E_2$ occurs during the decryption queries in **Query Phase 2**, there exits a hash collision that $T = H_2(C_1, C_2, C_3^*, C_6^*) = T^* = H_2(C_1^*, C_2^*, C_3^*, C_6^*)$. We have

$$Pr[E_2] \le \varepsilon_{CR},$$

  where $\varepsilon_{CR}$ is the probability that $\mathcal{A}$ successfully breaks the collision resistance of $H_2$.
- **The failures caused by simulation limits**: During the whole simulation process, some events will make it abort in which $\mathcal{B}$ cannot give a logical answer to decryption queries from $\mathcal{A}$.

$E_1$ and $E_3$ occur when the elements of queried ciphertexts satisfy the relation: $T + C_4 x_v + x_w = 0$. Because that element $T$ depends on the submitted ciphertext and $x_v, x_w$ are fixed, the probabilities of $E_1$ and $E_3$ depend on the randomness of $C_4 = r_0$ chosen from $Z_p$. It means that equation $T + C_4 x_v + x_w$ happens with a probability which is at most $1/p$ in one round query. We have

$$Pr[E_1 \vee E_3] \leq Pr[E_1] + Pr[E_3] \leq q_D/p,$$

where $q_D$ means $\mathcal{A}$ is allowed to issue *Decryption query* for $q_D$ times.

We obtain the failure and abortion probability $Pr[F]$ as

$$Pr[F] = Pr[E_1 \vee E_2 \vee E_3] \leq \varepsilon_{CR} + q_D/p.$$

Below we analyze the probability that $\mathcal{B}$ successfully guess the value of $b$. If $\mathcal{Z} = e(g,g)^{a^{q+1}s}$, the simulation provided by $\mathcal{B}$ is perfect, and in view of $\mathcal{A}$, the challenge ciphertext is the same as a real ciphertext. We have that $\Pr[b' = 0|b = 0] = (\frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}^{\text{IND-SAS-CCA}}(k)) \cdot Pr[\neg F]$; otherwise, which means $\mathcal{Z}$ is a random element of $\mathbb{G}_T$, the challenge ciphertext hides the message perfectly, and the probability that $\mathcal{A}$ correctly guesses the bit $\beta$ is $\frac{1}{2}$. Thus, the probability that $\mathcal{B}$ correctly guesses the bit $b$ is $(1 - \frac{1}{2} \cdot Pr[\neg F])$, e.g. $\Pr[b' = 1|b = 1] = 1 - \frac{1}{2} \cdot Pr[\neg F]$. Therefore, we have the followings.

$$\begin{aligned}
&Pr[b' = b] \\
&= Pr[b' = 0 \wedge b = 0] + Pr[b' = 1 \wedge b = 1] \\
&= \frac{1}{2}(Pr[b' = 0|b = 0] + Pr[b' = 1|b = 1]) \\
&= \frac{1}{2}\Big((\frac{1}{2} + \mathrm{Adv}_{\mathcal{A}}^{\text{IND-SAS-CCA}}(1^k)) \cdot Pr[\neg F] + (1 - \frac{1}{2} \cdot Pr[\neg F])\Big) \\
&= \frac{1}{2} + \frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{\text{IND-SAS-CCA}}(1^k) \cdot Pr[\neg F] \\
&\geq \frac{1}{2} + \frac{1}{2}\mathrm{Adv}_{\mathcal{A}}^{\text{IND-SAS-CCA}}(1^k) \cdot [1 - (\varepsilon_{CR} + q_D/p)],
\end{aligned}$$

where $[1 - (\varepsilon_{CR} + q_D/p)]$ is a non-negligible probability.

From the probability analysis above, we know that if $\mathcal{A}$ can break the IND-SAS-CCA security of CP-ABEET scheme with a non-negligible advantage, then $\mathcal{B}$ has a non-negligible advantage in solving the decisional $q$-BDHE problem. This completes the proof of Theorem 2. ∎

## VI. AN OUTSOURCED CONSTRUCTION IN THE STANDARD MODEL

The construction in Section IV addresses the aforementioned problems, but if Dec and Test algorithms are executed locally, the computational overhead is too high for resource-constrained clients; if these algorithms are executed by the server, there is a risk of data leakage.

To optimize the computation efficiency of our CP-ABEET scheme, we give an improved construction called *Outsourced CP-ABEET* (OCP-ABEET) that can be proven secure in standard model. We take advantage of outsourcing technique which was firstly proposed by Green *et al.* [25] and combine it with above basic CP-ABEET scheme. This new construction includes eight algorithms. Setup and Enc algorithms are defined as the same with the former construction defined in Section IV. KeyGen, Transform$_1$, Transform$_2$, Dec, Trapdoor and Test algorithms are defined as follows.

- KeyGen($\mathcal{SP}$, Msk, $S$): The key generation algorithm takes as input the system parameters $\mathcal{SP}$, the master secret key Msk and a set $S$ of attributes. Then it chooses random elements $z, z', t, t' \in \mathbb{Z}_p$ and computes:

$$\begin{aligned}
\mathsf{Sk}_Z &= (z, z'), \\
\mathsf{Sk}_S &= \big(K_d = h^{\alpha/z} g^{at/z}, L = g^{t/z}, \{K_x = h_x^{t/z}\}_{x \in S}, \\
K_t &= h^{\beta/z'} g^{at'/z'}, L' = g^{t'/z'}, \{K'_x = h_x^{t'/z'}\}_{x \in S}\big).
\end{aligned}$$

- Transform$_1$(Ct, Sk$_S$): Given a ciphertext Ct and a private key Sk$_S$, it partially decrypt Ct by the reconstruction property of LSSS. Suppose the attribute set $S$ can satisfy the access structure $(M, \rho)$ of Ct. Let $I \subseteq \{1, 2, \cdots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$ where $\rho$ belongs to $(M, \rho)$. We define the set $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of any secret $s$ according to $M$ of $(M, \rho)$, then we have $\sum_{i \in I} w_i \lambda_i = s$. It computes

$$\begin{aligned}
X &= e(C_3, K_d)/(\prod_{i \in I}(e(c_i, L)e(d_i, K_{\rho(i)}))^{w_i}) \\
&= e(g,h)^{\alpha s/z} e(g,g)^{ast/z}/(\prod_{i \in I} e(g,g)^{a\lambda_i w_i t/z}) \\
&= e(g,h)^{\alpha s/z} = e(g_1, h)^{s/z}, \\
X' &= e(C_3, K_t)/(\prod_{i \in I}(e(c_i, L')e(d_i, K'_{\rho(i)}))^{w_i}) \\
&= e(g,h)^{\beta s/z'} e(g,g)^{ast'/z'}/(\prod_{i \in I} e(g,g)^{a\lambda_i w_i t'/z'}) \\
&= e(g,h)^{\beta s/z'} = e(g_2, h)^{s/z'}.
\end{aligned}$$

  Then it outputs $(X, X')$.

- Dec(Ct, Sk$_Z$, Sk$_S$): The decryption algorithm computes $T' = H_2(C_1, C_2, C_3, C_6)$, and check whether $e(C_3, u^{T'} v^{C_4} w) = e(C_5, g)$. If the equation doesn't hold, output $\perp$; otherwise, it runs Transform$_1$(Ct, Sk$_S$) to get $(X, X')$. Then it computes

$$\hat{H} = \frac{C_2}{(X')^{z'}} \quad \text{and} \quad \hat{m} = \frac{C_1}{(X)^z},$$

  and outputs $\hat{m}$ if the following equation hold:

$$\hat{H} = H_1(\hat{m}).$$

- Trapdoor($\mathcal{SP}$, Msk, $S$): The trapdoor algorithm takes as input the system parameters $\mathcal{SP}$, the master secret key Msk and the set $S$ of attributes. Then it chooses random elements $\hat{z}, \hat{t} \in \mathbb{Z}_p$, computes and outputs trapdoor (Td$_Z$, Td$_S$) as:

$$\begin{aligned}
\mathsf{Td}_Z &= \hat{z}, \\
\mathsf{Td}_S &= (\hat{K} = h^{\beta/\hat{z}} g^{a\hat{t}/\hat{z}}, \hat{L} = g^{\hat{t}/\hat{z}}, \{\hat{K}_x = h_x^{\hat{t}/\hat{z}}\}_{x \in S}).
\end{aligned}$$

- Transform$_2$(Ct, Td$_S$): It takes a ciphertext Ct and a trapdoor Td$_S$ as input. Suppose the attribute set $S$ can satisfy

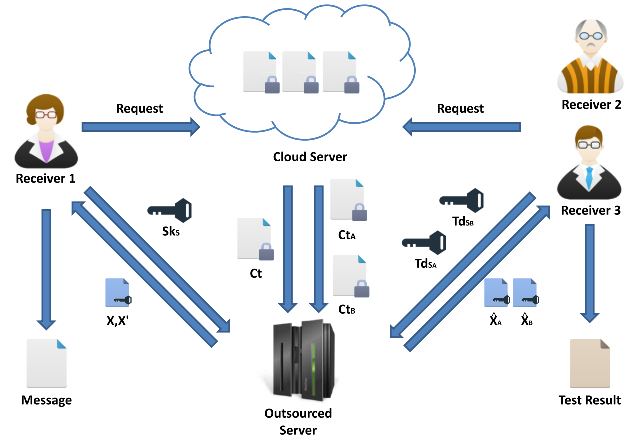


**FIGURE 2.** Flowchart of OCP-ABEET.

the access structure $(M, \rho)$ of $\mathsf{Ct}$. Let $I \subseteq \{1, 2, \cdots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$ where $\rho$ belongs to $(M, \rho)$. We define the set $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that if $\{\lambda_i\}$ are valid shares of any secret $s$ according to $M$ of $(M, \rho)$, then we have $\sum_{i \in I} w_i \lambda_i = s$. Then it computes

$$\begin{aligned}\hat{X} &= e(C_3, \hat{K})/(\prod_{i \in I}(e(c_i, \hat{L})e(d_i, \hat{K}_{\rho(i)}))^{w_i}) \\ &= e(g, h)^{\beta s/\hat{z}} e(g, g)^{as\hat{t}/\hat{z}}/(\prod_{i \in I} e(g, g)^{a\lambda_i w_i \hat{t}/\hat{z}}) \\ &= e(g, h)^{\beta s/\hat{z}} = e(g_2, h)^{s/\hat{z}}.\end{aligned}$$

- $\mathsf{Test}(\mathsf{Ct}_A, \mathsf{Td}_{Z_A}, \mathsf{Td}_{S_A}, \mathsf{Ct}_B, \mathsf{Td}_{Z_B}, \mathsf{Td}_{S_B})$: It runs $\mathsf{Transform}_2(\mathsf{Ct}_A, \mathsf{Td}_{S_A})$ and $\mathsf{Transform}_2(\mathsf{Ct}_B, \mathsf{Td}_{S_B})$ to get $\hat{X}_A$ and $\hat{X}_B$. Then it computes

$$H_A = \frac{C_{2_A}}{(\hat{X}_A)^{\mathsf{Td}_{Z_A}}} = \frac{C_{2_A}}{(\hat{X}_A)^{\hat{z}_A}},$$
$$H_B = \frac{C_{2_B}}{(\hat{X}_B)^{\mathsf{Td}_{Z_B}}} = \frac{C_{2_B}}{(\hat{X}_B)^{\hat{z}_B}}.$$

  Finally it outputs 1 if the equation $H_A = H_B$ holds, and 0 otherwise.

In our construction, users generate trapdoors based on their attribute sets and send them to the outsourced server for transforming ciphertexts. As a result, most of the computational costs of DEC and TEST are transferred to the outsourced server. Concretely, the main operations in $\mathsf{Dec}$ algorithm and $\mathsf{Test}$ algorithm are split into two algorithms, $\mathsf{Transform}_1$ and $\mathsf{Transform}_2$, respectively, which are outsourced to the third-party servers. After the outsourced server returns the transformation result, the user can quickly complete the final steps of decryption or equality test. It is ensured that the outsourced server does not learn information about the messages. Figure 2 shows the outsourcing framework of CP-ABEET.

Our OCP-ABEET scheme also achieves OW-SAS-CCA and IND-SAS-CCA security in standard model. The correctness and security can be proven by combining corresponding proofs of the CP-ABEET scheme above.

## VII. EFFICIENCY EVALUATION

We compare our CP-ABEET scheme with some related schemes in Table 1, in terms of computational complexity, functional properties, assumptions, security level and etc. In the comparison we mainly consider the dominant computation, e.g. bilinear pairing evaluation and exponentiation operation, in encryption, decryption and test algorithms. The second to the fourth columns show the computation costs of $\mathsf{Enc}$, $\mathsf{Dec}$ and $\mathsf{Test}$ algorithms. The fifth column indicates whether the scheme is attribute-based. The sixth column shows the authorization type of each scheme. The following two columns indicate the underlying assumptions and security levels of the related schemes. The last column shows whether the scheme is proven secure in ROM (Random oracle model) or SM (Standard model).

From Table 1, we can know that our CP-ABEET and OCP-ABEET schemes enjoy the highest level of security guarantee among all the attribute-based encryption schemes supporting equality test. And our OCP-ABEET scheme provides almost the best efficiency among all the ABEET schemes.

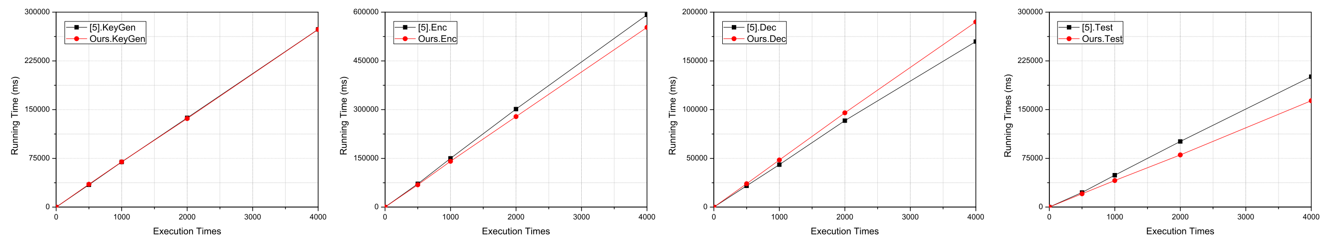
To better show the practical performance of our new CP-ABEET, we strictly simulated our scheme system and made a practical comparison with the last CP-ABEET [10] scheme which is proven secure in random oracle. We mainly used the Java Pairing-Based Cryptography (JPBC) library and the Bouncycastle library to realize our system. And all of these experiments were executed by Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz on Windows 7 64-bit system with 8GB memory. We ran the complete system and obtained the running time of main algorithms: $\mathsf{KeyGen}$, $\mathsf{Enc}$, $\mathsf{Dec}$ and $\mathsf{Test}$ algorithms. To make the result more universal and credible, we independently set the test times as 500, 1000, 2000 and 4000. Figure 3(a), 3(b), 3(c), 3(d) show us that our CP-ABEET is more efficient than scheme in [10] in $\mathsf{Test}$ algorithm, and the efficiency of $\mathsf{KeyGen}$, $\mathsf{Enc}$ and $\mathsf{Dec}$ algorithm is similar with that in [10]. As the number of tests increases, the running time increases linearly.

To illustrate the efficiency of our OCP-ABEET scheme, we also implemented it and compared the computational cost between our first CP-ABEET scheme and our OCP-ABEET scheme. As shown in Figure 4(a), 4(b), 4(c) and 4(d), the black line represents the computational cost of our OCP-ABEET scheme, the red line represents local computing portion in our OCP-ABEET scheme and the blue line represents outsourcing portion in our OCP-ABEET scheme (mainly the computational cost of $\mathsf{Transform}_1$ and $\mathsf{Transform}_2$ algorithms). To support outsourced computing, $\mathsf{KeyGen}$ algorithm of our OCP-ABEET scheme has a slightly higher computational cost, while the two $\mathsf{Enc}$ algorithms have equivalent computational cost. Surprisingly, since most of the computations are outsourced to the outsourced server, the result of $\mathsf{Dec}$ algorithm and $\mathsf{Test}$ algorithm can be obtained by performing simple calculation locally. While ensuring security, it is convenient for devices

**TABLE 1. Performance comparison with related schemes.**

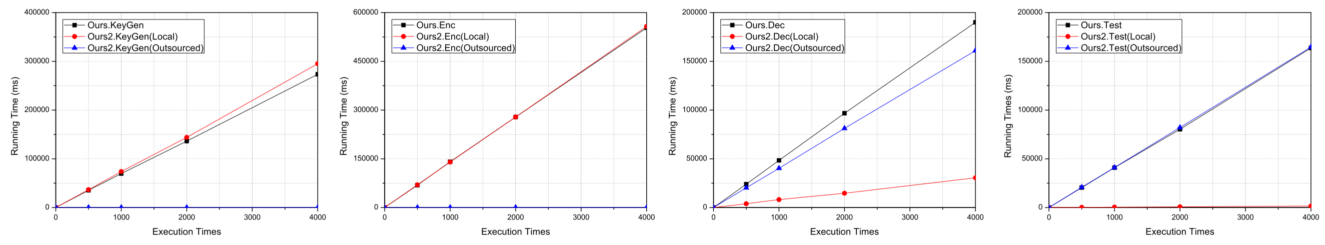
| | KP-ABEwET [8] | CP-ABE-ET [9] | CP-ABEET [10] | Our CP-ABEET | Our OCP-ABEET |
|---|---|---|---|---|---|
| KeyGen | $2A_uE$ | $(4+6A_u+12A_u^2)E$ | $(4+2A_u)E$ | $(4+2A_u)E$ | $(6+2A_u)E$ |
| Enc | $(2A_u+3)E$ | $(2N_U+11)E$ | $(3\ell+5)E+2P$ | $(3\ell+6)E$ | $(3\ell+6)E$ |
| Dec | $(2A_u+2)E+2A_uP$ | $(8A_u+6)E+12P$ | $(2A_u+2)E+(4A_u+2)P$ | $(2A_u+2)E+(4A_u+4)P$ | $4E+2P$ |
| Test | $2A_uE+2A_uP$ | $(8A_u+4)E+14P$ | $A_uE+(2A_u+3)P$ | $A_uE+(2A_u+1)P$ | $2E$ |
| $\mathsf{Ct}_{size}$ | $(4+2A_u)\lvert\mathbb{G}\rvert+2\lvert\mathbb{Z}_p\rvert$ | $8\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$ | $(4+2\ell)\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$ | $(2+2\ell)\lvert\mathbb{G}\rvert+2\lvert\mathbb{G}_T\rvert+\lvert\mathbb{Z}_p\rvert$ | $(2+2\ell)\lvert\mathbb{G}\rvert+2\lvert\mathbb{G}_T\rvert+\lvert\mathbb{Z}_p\rvert$ |
| $\mathsf{Sk}_{size}$ | $2A_u\lvert\mathbb{G}\rvert$ | $(4+6A_u)\lvert\mathbb{G}\rvert$ | $(4+2A_u)\lvert\mathbb{G}\rvert$ | $(4+2A_u)\lvert\mathbb{G}\rvert$ | $(4+2A_u)\lvert\mathbb{G}\rvert+2\lvert\mathbb{Z}_p\rvert$ |
| **Attibute-Based** | Yes | Yes | Yes | Yes | Yes |
| **Outsourced** | No | No | No | No | Yes |
| **Authorization** | Flexible | Flexible | Flexible | Flexible | Flexible |
| **Assumption** | tDBDH | DLIN | Decisional $q$-BDHE | Decisional $q$-BDHE | Decisional $q$-BDHE |
| **Security** | OW-CCA & T-CCA | IND-ID-CPA | OW-SAS-CCA & IND-SAS-CCA | OW-SAS-CCA & IND-SAS-CCA | OW-SAS-CCA & IND-SAS-CCA |
| **ROM/SM** | ROM | SM | ROM | SM | SM |

T-CCA [8]: testability against chosen-ciphertext attack of authorization under the chosen sets of attributes. $N_U$ is the amount of attributes in Wang et al.'s system [9]. We use $A_u$ to denote the number of attributes used in KeyGen, Enc, Dec and Test algorithms, and use $|\mathbb{G}|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ to denote the element size of $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. In CP-ABEET and OCP-ABEET schemes, $\ell$ is the number of rows of the access matrix $M$. ROM means the scheme is proven secure in random oracle model while SM means the scheme is proven secure in standard model. Both the IND-ID-CPA model in [9] and OW-SAS-CCA and IND-SAS-CCA models in CP-ABEET and OCP-ABEET schemes consider the selective access structure security, in which the adversary submits its challenge access structure before seeing the public parameters.



(a) Comparison of KeyGen (b) Comparison of Enc (c) Comparison of Dec (d) Comparison of Test

**FIGURE 3. Computational costs of KeyGen, Enc, Dec and Test algorithms in [10] and our scheme.**



(a) Comparison of KeyGen (b) Comparison of Enc (c) Comparison of Dec (d) Comparison of Test

**FIGURE 4. Computational costs of KeyGen, Enc, Dec and Test algorithms in our schemes.**

with limited computing power, such as mobile phones, to execute our OCP-ABEET scheme.

## VIII. CONCLUSION

In this paper, we propose a new construction of CP-ABEET scheme which is proven secure in standard model. Our CP-ABEET scheme supports flexible authorized equality test on ciphertext. One-wayness is achieved if the adversary is given trapdoor and indistinguishability is achieved if the adversary is not given trapdoor. This scheme can be applied to delete the flexible authorized deduplication on encrypted data, which means users can optimize the storage space in cloud by delegating their equality test. By the comparison with related works, we achieve a more secure CP-ABEET scheme in standard model. In addition, our OCP-ABEET scheme in standard model is more efficient for users with low computing capability and mobile devices.

## REFERENCES

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Adv. Cryptol.-EUROCRYPT Int. Conf. Theory Appl. Cryptograph. Techn.*, Interlaken, Switzerland, May 2004, pp. 506–522.
[2] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2000, pp. 44–55.
[3] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Proc. Int. Conf. Topics Cryptol.*, 2010, pp. 119–131.
[4] Q. Tang, "Towards public key encryption scheme supporting equality test with fine-grained authorization," in *Proc. Inf. Secur. Privacy-16th Australas. Conf. (ACISP)*, Melbourne, Australia, Jul. 2011, pp. 389–406.
[5] Q. Tang, "Public key encryption schemes supporting equality test with authorisation of different granularity," *Int. J. Appl. Cryptogr.*, vol. 2, no. 4, p. 304, 2012.

[6] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," *Secur. Commun. Netw.*, vol. 5, no. 12, pp. 1351–1362, Dec. 2012.

[7] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 458–470, Mar. 2015.

[8] H. Zhu, L. Wang, H. Ahmad, and X. Niu, "Key-policy attribute-based encryption with equality test in cloud computing," *IEEE Access*, vol. 5, pp. 20428–20439, 2017.

[9] 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*, vol. 6, pp. 760–771, 2018.

[10] Y. Cui, Q. Huang, J. Huang, H. Li, and G. Yang, "Ciphertext-policy attribute-based encrypted data equality test and classification," *Cryptol. ePrint Arch.*, to be published.

[11] S. Ma, M. Zhang, Q. Huang, and B. Yang, "Public key encryption with delegated equality test in a multi-user setting," *Comput. J.*, vol. 58, no. 4, pp. 986–1002, Apr. 2015.

[12] K. Zhang, J. Chen, H. T. Lee, H. Qian, and H. Wang, "Efficient public key encryption with equality test in the standard model," *Theor. Comput. Sci.*, vol. 755, pp. 65–80, Jan. 2019.

[13] Y. Ling, S. Ma, Q. Huang, X. Li, and Y. Ling, "Group public key encryption with equality test against offline message recovery attack," *Inf. Sci.*, vol. 510, pp. 16–32, Feb. 2020.

[14] Y. Wang, Q. Huang, H. Li, J. Huang, G. Yang, and W. Susilo, "Public key authenticated encryption with designated equality test and its applications in diagnostic related groups," *IEEE Access*, vol. 7, pp. 135999–136011, 2019.

[15] S. Ma, "Identity-based encryption with outsourced equality test in cloud computing," *Inf. Sci.*, vol. 328, pp. 389–402, Jan. 2016.

[16] T. Wu, S. Ma, Y. Mu, and S. Zeng, "ID-based encryption with equality test against insider attack," in *Proc. Inf. Secur. Privacy-22nd Australas. Conf. (ACISP)*, Auckland, New Zealand, Jul. 2017, pp. 168–183.

[17] L. Wu, Y. Zhang, K.-K.-R. Choo, and D. He, "Efficient identity-based encryption scheme with equality test in smart city," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 1, pp. 44–55, Jan. 2018.

[18] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2005, pp. 457–473.

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

[20] N. Attrapadung, B. Libert, and E. D. Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *Proc. Int. Conf. Pract. Theory Public Key Cryptogr. Conf. Public Key Cryptogr.*, 2011, pp. 90–108.

[21] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 11, pp. 2150–2162, Nov. 2012.

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

[23] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proc. Int. Colloq. Automata, Lang., Program.*, 2008, pp. 579–591.

[24] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC* (Lecture Notes in Computer Science). 2008, pp. 321–334.

[25] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Secur. Symp.*, San Francisco, CA, USA, Aug. 2011, pp. 1–16.

[26] Y. Cui, Q. Huang, J. Huang, H. Li, and G. Yang, "Outsourced ciphertext-policy attribute-based encryption with equality test," in *Proc. Inf. Secur. Cryptol.-14th Int. Conf. (INSCRYPT)*, Fuzhou, China, Dec. 2018, pp. 448–467.

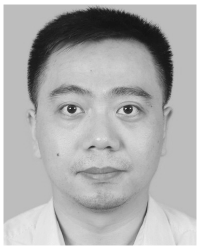
[27] J. Lai, R. H. Deng, S. Liu, and W. Kou, "Efficient CCA-secure PKE from identity-based techniques," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 5985, San Francisco, CA, USA, Mar. 2010, pp. 132–147.

[28] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Conf. Pract. Theory Public Key Cryptogr.*, Taormina, Italy, Mar. 2011, pp. 53–70.

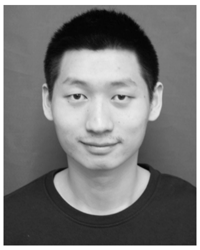
[29] A. Beimel, "Secure schemes for secret sharing and key distribution using Pell's equation," *Int. J. Pure Appl. Math.*, vol. 85, no. 5, pp. 933–937, 1996.

**YUANHAO WANG** received the B.S. degree from South China Agricultural University, in 2017, where he is currently pursuing the M.S. degree with the College of Mathematics and Informatics. His research interests include information security, searchable encryption, and digital signature.

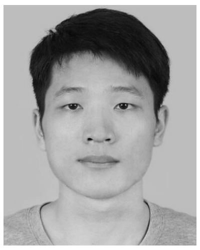
**YUZHAO CUI** received the B.S. and M.S. degrees from South China Agricultural University, in 2016 and 2019, respectively. His research interests include information security, searchable encryption, and digital signature.



**QIONG HUANG** (Member, IEEE) received the Ph.D. degree from the City University of Hong Kong, in 2010. He is currently a Professor with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China, and with the Director of Guangzhou Key Laboratory of Intelligent Agriculture, Guangzhou. He has published more than 100 research articles in international conferences and journals. His research interests include cryptography and information security, in particular, cryptographic protocol design and analysis. He served as a program committee member for many international conferences.



**HONGBO LI** received the B.S. and M.S. degrees from South China Agricultural University, Guangzhou, China, where he is currently pursuing the Ph.D. degree with the College of Mathematics and Informatics. His research interests include applied cryptography and cloud security.



**JIANYE HUANG** received the B.S. and M.S. degrees from South China Agricultural University. He is currently pursuing the Ph.D. degree with the School of Computing and Information Technology, University of Wollongong, NSW, Australia. His research interests include applied cryptography and cloud security.

**GUOMIN YANG** (Senior Member, IEEE) received the bachelor's, master's, and Ph.D. degrees in computer science from the City University of Hong Kong, in 2004, 2006, and 2009, respectively. In 2012, he was a Research Scientist with Temasek Laboratories, National University of Singapore. He is currently an Associate Professor with the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology (iC2), University of Wollongong (UOW), Australia. His research interests are applied cryptography and network security. He received the Australian Research Council Discovery Early Career Researcher Award, in 2015.