Received January 27, 2019, accepted February 11, 2019, date of publication February 15, 2019, date of current version March 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2899680

Authorized Equality Test on Identity-Based Ciphertexts for Secret Data Sharing via Cloud Storage

HONGBO LI[1], QIONG HUANG[1], (Member, IEEE), SHA MA[1], JIAN SHEN[2], (Member, IEEE), AND WILLY SUSILO[3], (Senior Member, IEEE)

[1]College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[2]School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, 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 Pearl River Nova Program of Guangzhou under Grant 201610010037, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2014A030306021, in part by the National Natural Science Foundation of China under Grant 61872152 and Grant 61872409, in part by the Guangdong Program for Special Support of Top-Notch Young Professionals under Grant 2015TQ01X796, and in part by the Graduate Student Overseas Study Program of South China Agricultural University under Grant 2018LHPY025.

ABSTRACT With the higher rate of using cloud storage, protecting data privacy becomes an important issue. The most effective solution is to encrypt data before uploading to the cloud. However, how to efficiently search over data encrypted with different keys is still an open problem. To address this problem, we introduce a new notion of the identity-based encryption with equality test supporting flexible authorization (IBEET-FA). It supports the test of whether two ciphertexts encrypted under the different keys encapsulate the same message, and in the meanwhile supports fine-grained authorization of the test. Based on the equality test on ciphertexts, there is a direct way to support an authorized user to search over ciphertexts of different users, which accelerates secret data sharing among a group of users. Besides, IBEET-FA does not suffer from the complex key management problem of its counterpart in the traditional public key infrastructure. We propose a concrete construction of IBEET-FA and prove it to be securely based on simple mathematical assumptions. The experimental results show that our IBEET-FA scheme is efficient and can satisfy various types of search over encrypted data.

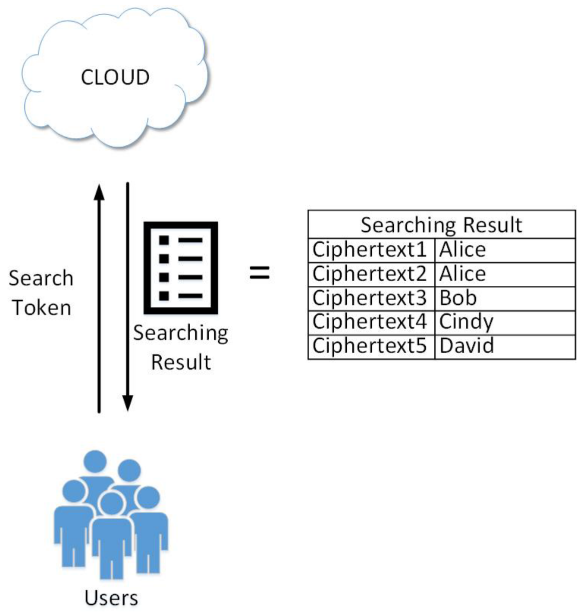
INDEX TERMS Equality test, identity-based encryption, flexible authorization, cloud storage, data sharing.

I. INTRODUCTION

Cloud computing is becoming increasingly popular in recent decades [1], [2]. Among all the cloud services, cloud storage is receiving a growing number of attention and applications [3]. The cloud service provider (CSP) provides large storage-space for users [4]. Users can outsource their data to the cloud and access the data via any network-connected device at any time, which saves local storage space and reduces data management burden. However, there are concerns about the leakage of data and user privacy. Even in a private cloud, there also are some concerns about data leakage in case of some adversary obtaining the password of the cloud manager and breaking into the cloud. Encrypting data before outsourcing is the most effective method to solve this problem. Let all data in the cloud be encrypted with different encryption keys. Even if an adversary broke into the cloud and obtained one or some decryption keys of the encrypted data, other data will be protected from leakage. Furthermore, in the real world, it is hard for an adversary to obtain these decryption keys. Unfortunately, many traditional operations on plaintexts cannot be directly applied to ciphertexts. For example, searching function over encrypted data is not available while using basic encryption schemes.

Consider the following scenario illustrated in Figure 1. In an enterprise, there is a group of employees who are working for a secret project. To protect the business secret, data sent to the employees are encrypted and stored into a

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



Searching Result	
Ciphertext1	Alice
Ciphertext2	Alice
Ciphertext3	Bob
Ciphertext4	Cindy
Ciphertext5	David

FIGURE 1. Search over ciphertexts encrypted with different encryption keys.

cloud. Usually, the encrypted data are encrypted with different encryption keys and may contain any kind of data, e.g. images, contract documents, videos, etc. Generally, only the one who holds the decryption key can decrypt the corresponding encrypted data. However, it obstructs data sharing among employees in the group. For example, Alice (in the group) needs files related to a specific keyword *w* for business purposes. It is hard to find these files from the cloud because data are encrypted with different keys. Besides, it is not easy for Alice to know who owns the files she wants to access. Hence, it is desirable to have a way to search over these ciphertexts. Furthermore, employees should have different levels of permission to search over the ciphertexts, because different employees have different authority.

Searchable symmetric encryption (SSE) [5] and public key encryption with keyword search (PEKS) [6] can support searching function on ciphertexts, however, they cannot feed the needs of the aforementioned scenario, as they only support the search over encrypted data of a single user. Public key encryption with equality test (PKEET) [7] is a new primitive which supports the test of whether two ciphertexts encrypted with different public keys encapsulate the same message without decryption. While anyone could do the test without permission in PKEET, it is not appropriate in the aforementioned scenario. A recent variant of PKEET called public key Encryption with equality test supporting flexible authorization (PKEET-FA) [8], better suits the scenario above, in which ciphertexts cannot be tested without authorization. In PKEET-FA, an authorized user can check whether two ciphertexts are decrypted the same plaintext without decryption even if the two ciphertexts are encrypted with different encryption keys. Hence, it can be transformed into search progress if we set one of the two ciphertexts as a search token. Furthermore, PKEET-FA supports four types of authorization, which is more suitable for the aforementioned scenario.

However, PKEET-FA works in the traditional Public Key Infrastructure (PKI) and suffers from the complex problem of public key management. Each user has to obtain a digital certificate on its public key from a certification authority (CA) before using the key. The issue, verification, management and revocation of certificates requires much effort. Identity-Based Cryptography (IBC) was introduced by Shamir [9] to solve this problem. In IBC, there is a trusted third party called the Key Generation Center (KGC), which is in charge of generating user secret keys according to their identities. Each user can simply use its identity as the public key, for example, email address, IP address and etc, and obtain its secret key from the KGC via a secure channel. People can send an encrypted message to a user even before the user gets its secret key. Due to the advantage of IBC, it is desirable and meaningful to transfer PKEET-FA to the identity-based setting.

A. OUR CONTRIBUTIONS

In order to support searching over ciphertexts encrypted with different public keys, in this paper, we introduce the notion of identity-Based encryption with equality test supporting flexible authorization (IBEET-FA), which is a variant of PKEET-FA in the identity-based setting. In IBEET-FA, ciphertexts are encrypted with the receiver's identity, and an authorized user can do the equality test on the ciphertexts. Furthermore, IBEET-FA does not suffer from complex key management as PKEET-FA.

We formally define IBEET-FA and present its security models. Roughly, IBEET-FA considers two security requirements, one-wayness and indistinguishability. The former says that if the adversary has the trapdoor, it cannot recover the message from a given ciphertext. The latter says that if the adversary is not given the trapdoor, it cannot distinguish which message the given ciphertext contains.

To demonstrate the new notion, we present a concrete construction of IBEET-FA from bilinear pairings. The construction is communicationally efficient in the sense that all the keys and trapdoors in the construction are short, each containing a small constant number of group elements. The construction is also computationally efficient, and experimental results show that its computational costs are almost the same as its counterpart in the PKI setting [8]. We prove that our IBEET-FA scheme is secure under the given models based on standard number-theoretic assumptions.

As far as we know, there is no efficient transform converting public key encryption (PKE) to identity-based encryption (IBE). It is not a trivial task to build an IBEET-FA scheme, although there is already a PKEET-FA construction in [8]. The difficulty lies in how to integrate the four types of authorization into an IBE scheme.

B. RELATED WORKS

Searchable encryption can be divided into two groups: symmetric searchable encryption [10] and public key searchable encryption [6]. The topic of this paper belongs to the latter.

The first searchable encryption scheme in public key settings (denoted by PEKS) was proposed by Boneh *et al.* [6] in 2004. In PEKS, people can share their private data with friends, by encrypting the data under the friend's public key. The friend can use its secret key to compute the trapdoor of a keyword, and use the trapdoor to search over the encrypted data. Since the introduction of PEKS, many variants have been proposed. For example, some works focus on fuzzy keyword search [11]–[13], some on flexible keyword search [14], and some on trapdoor privacy [15]–[18]. In 2010, Yang *et al.* [7] proposed the notion of public key encryption with equality test (PKEET), which allows a user to directly compare whether two ciphertexts have the same message. However, since anyone can do the comparison without authorization, it may leak certain private information about the data. Hence, researchers have studied how to construct fine-grained authorization mechanisms in PKEET. To name a few, Tang [19] proposed a PKEET scheme supporting authorization, in which a test server can only perform an equality test on ciphertexts encrypted with public keys of two specific users. The restriction is due to that users have to secretly share a tuple of session keys with each other during authorization. To address this issue, Tang [20] proposed another PKEET scheme called AoN-PKEET, which supports user-level authorization. In AoN-PKEET, once the test server obtains trapdoors from users, it will be able to test all the ciphertexts encrypted with these users' public keys. Ma *et al.* [8] proposed a PKEET scheme with flexible authorization, called PKEET-FA, which supports four types of authorization, i.e. user-level authorization, ciphertext level, user-specific ciphertext level and ciphertext-to-user level authorizations (c.f. Fig. 2). To improve the computational efficiency, Lin *et al.* [21] proposed another PKEET-FA scheme which does not use bilinear pairings. Xu *et al.* [22] introduced the notion of verifiable PKEET, in which a user can check whether the server performs the authorized equality test honestly or not. Canard *et al.* [23] introduced a new notion related to PKEET, called plaintext-checkable encryption, in which it is universally possible to check whether a given ciphertext is the encryption of a given plaintext under the key.

Ma [24] proposed the first identity-based encryption scheme supporting equality test (IBEET) which is a combination of identity-based encryption and PKEET and simplifies the complex certificate management in PKEET. However, their scheme only supports the user-level authorization, and the test server can perform the test on all ciphertexts of users who issues trapdoors to the server. To the best of our knowledge, currently, there is no IBEET scheme in the literature supporting flexible and fine-grained authorization.

We notice that the IBEET scheme in [24] does not achieve one-wayness as claimed. Roughly speaking, after obtaining trapdoor td_{ID_t} of the challenge identity ID_t, the adversary re-randomizes the challenge ciphertext and submits the new ciphertext to the decryption oracle to obtain the challenge message, thus breaking the one-wayness. Hence, it is non-trivial to build a secure IBEET-FA scheme.

To support a specified group of users to perform the equality test on ciphertexts., Zhu *et al.* [25] and Wang *et al.* [26], respectively, proposed key-policy and ciphertext-policy attribute-based encryption with equality test schemes.

C. ORGANIZATION

In the next section, we give a brief description of some preliminaries. In Section III we give the definition of IBEET-FA and propose an instance in Section IV. The security model and security analysis is provided in Section V. We compare our scheme with the PKEET-FA scheme [8] and present the experiment results in Section VI. The paper is concluded in Section VII.

II. PRELIMINARIES

A. BILINEAR PAIRING

Bilinear pairing is a mathematics tool which plays an important role in public key cryptography [27]. It is a one-way mapping from a cyclic group $\mathbb{G}_1$ to another cyclic group $\mathbb{G}_T$, e.g. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where both $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of prime order p. It should satisfy the following properties.

- **Bilinearity:** For any $g, h \in \mathbb{G}_1$ and any $a, b \in \mathbb{Z}_p$, it holds that $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{a\cdot b}$.
- **Non-degeneracy:** For any generator $g \in \mathbb{G}_1$, $\hat{e}(g, g)$ should not be equal to the identity element of $\mathbb{G}_T$.
- **Computability:** For any $g, h \in \mathbb{G}_1$, $\hat{e}(g, h)$ can be computed in polynomial time.

B. MATHEMATICAL ASSUMPTIONS

1) BILINEAR DIFFIE-HELLMAN ASSUMPTION

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be cyclic groups with the same prime order p. Let $g \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear pairing. Given $g, g^x, g^y, g^z \in \mathbb{G}_1$, where x, y, z are randomly chosen from $\mathbb{Z}_p$, the Bilinear Diffie-Hellman (BDH) problem [28], [29] is to compute $\hat{e}(g, g)^{xyz}$. The Bilinear Diffie-Hellman assumption says that for any PPT adversary $\mathcal{A}$, its advantage

$$\mathrm{Adv}_{\mathcal{A}}^{BDH}(k) \overset{\text{def}}{=} \Pr[\mathcal{A}(g, g^x, g^y, g^z) = e(g, g)^{xyz}]$$

is negligible in the security parameter k.

2) DECISIONAL BILINEAR DIFFIE-HELLMAN ASSUMPTION

Let $\mathbb{G}_1, \mathbb{G}_T, \hat{e}, p, g$ be as above. Given $g^x, g^y, g^z \in \mathbb{G}_1$ and $Z \in \mathbb{G}_T$, where x, y, z are randomly selected from $\mathbb{Z}_p$, the Decisional Bilinear Diffie-Hellman (DBDH) problem [28], [29] is to decide if $Z = \hat{e}(g, g)^{xyz}$ or $Z = \hat{e}(g, g)^r$ for a random $r \in \mathbb{Z}_p$. The Decisional Bilinear Diffie-Hellman assumption says that for any PPT adversary $\mathcal{A}$, its advantage

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}}^{DBDH}(k) \overset{\text{def}}{=} & |\Pr[\mathcal{A}(g, g^x, g^y, g^z, \hat{e}(g,g)^{xyz}) = 1] \\ & - \Pr[\mathcal{A}(g, g^x, g^y, g^z, \hat{e}(g,g)^{r}) = 1]|\end{aligned}$$

is negligible in the security parameter k.

3) DISCRETE LOGARITHM ASSUMPTION

Let $\mathbb{G}$ be a cyclic group with prime order p. Let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$ and x is a random element of $\mathbb{Z}_p$. Given $\mathbb{G}, p, g, g^x$, the Discrete Logarithm (DL) problem [30], [31] is to compute the discrete logarithm x. The Discrete Logarithm assumption says that for any PPT adversary $\mathcal{A}$, its advantage

$$\mathrm{Adv}_{\mathcal{A}}^{DL}(k) \overset{\text{def}}{=} \Pr[\mathcal{A}(\mathbb{G}, p, g, g^x) = x]$$

is negligible in the security parameter k.

III. IBEET-FA DEFINITION

In this section we present the system model and the algorithm definitions of IBEET-FA.

A. SYSTEM MODEL

There are three parties in the system model of IBEET-FA, i.e. a KGC, a cloud, a trusted proxy and a group of users. Firstly, every user has one or more identities, e.g. email address, staff number or phone number, etc.. In one round communication, there should be two users act as a sender and a receiver, respectively. For instance, the sender encrypts a keyword utilizing the Encrypt algorithm of the IBEET-FA scheme, then uploads the ciphertext to the cloud. After that, the receiver can download and decrypt the ciphertext to obtain the encrypted keyword. In another round of communication, there may be another sender and another receiver. Using the same manner above, the new sender sends an encrypted keyword to the new receiver via the cloud. All the ciphertexts are stored in the cloud and accessible to both of the two receivers. After the two round communication, we assume the two receivers need to compare whether the keyword they received are same or not and the limitation is that the two receivers cannot expose their keywords to each other. In this case, the two can run the Aut algorithms to the proxy authorized trapdoors, thus the proxy can run the Test algorithm to check the equality of the two encrypted keywords without decryption. Note that, it directly is a searching method if one of the two trapdoors is a search token. To support different level of permission, in IBEET-FA, there are four types of authorized trapdoors and corresponding Test algorithms. Figure 2 describes the following four types of authorizations.

Type-1(User level authorization.) All ciphertexts of the user can be compared with any ciphertext of any user.

Type-2(Ciphertext level authorization.) A specific ciphertext of the user can be compared with some ciphertext of any user.

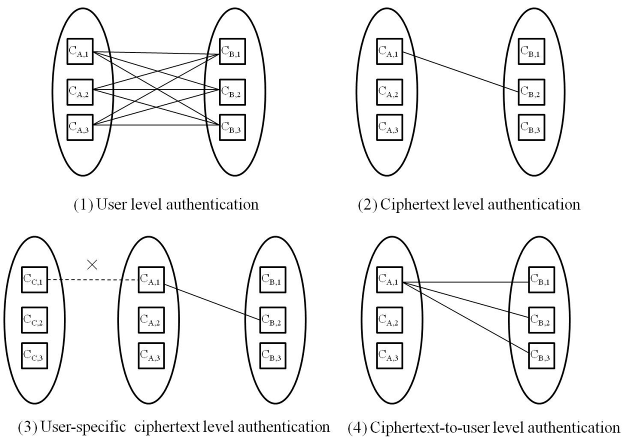

$C_{A,1}$
$C_{A,2}$
$C_{A,3}$
$C_{B,1}$
$C_{B,2}$
$C_{B,3}$
$C_{C,1}$
$C_{C,2}$
$C_{C,3}$
×
(1) User level authentication
(2) Ciphertext level authentication
(3) User-specific ciphertext level authentication
(4) Ciphertext-to-user level authentication

FIGURE 2. Different authorizations of ciphertexts in PKEET-FA [8]. (a) User level authorization. (b) Ciphertext level authorization. (c) User-specific ciphertext level authorization. (d) Ciphertext-to-user level authorization.

Type-3(User-specific ciphertext level authorization.) A specific ciphertext of the user can be compared with some ciphertext of a specific user.

Type-4(Ciphertext-to-user level authorization.) A specific ciphertext of the user can be compared with any other ciphertext. It is a combination of Type-1 authorization and Type-2 authorization.

B. ALGORITHMS

Definition 1 (IBEET-FA): An identity-based encryption with equality test supporting flexible authorization (IBEET-FA) consists of the following probabilistic polynomial-time (PPT) algorithms.

1) Setup(1^k): Given the security parameter k as input, the algorithm outputs the global parameter $\mathcal{K}$.
2) KeyGen$(\mathcal{K})$: Given the global parameter $\mathcal{K}$ as input, the algorithm outputs the master secret key msk and master public key mpk.
3) Extract(ID, msk): The algorithm takes as input a user's identity ID and the master secret key msk, and outputs the user's secret key dk_{ID}.
4) Encrypt(M, ID, mpk): The algorithm takes as input a message M, a user's identity ID and the master public key mpk, and outputs a ciphertext C.
5) Decrypt$(C, ID, \mathsf{dk}_{ID}, \mathsf{mpk})$: The algorithm takes as input a ciphertext C, a user's identity ID, its secret key dk_{ID} and the master public key mpk, and outputs a message M or $\perp$ indicating decryption failure.

Type-1 (Authorization):

6) Aut$_1(ID_i, \mathsf{dk}_{ID_i})$: Given a user's identity ID_i and its secret key dk_{ID_i}, the algorithm outputs a Type-1 trapdoor $td_{(i,1)}$.
7) Test$_1(C_i, td_{(i,1)}, C_j, td_{(j,1)})$: The algorithm takes as input two ciphertexts (C_i, C_j) and two Type-1 trapdoors $(td_{(i,1)}, td_{(j,1)})$, and outputs 1 if C_i and C_j contain the same message and 0 otherwise.

Type-2 Authorization:

8) $\mathsf{Aut}_2(ID_i, \mathsf{dk}_{ID_i}, C_i)$: Given a user's identity ID_i, its secret key dk_{ID_i} and a ciphertext C_i, the algorithm outputs a Type-2 trapdoor $td_{(i,2,C_i)}$.
9) $\mathsf{Test}_2(C_i, td_{(i,2,C_i)}, C_j, td_{(j,2,C_j)})$: The algorithm takes as input two ciphertexts (C_i, C_j) and two Type-2 trapdoors $(td_{(i,2,C_i)}, td_{(j,2,C_j)})$, and outputs 1 if C_i and C_j contain the same message and 0 otherwise.

Type-3 Authorization:

10) $\mathsf{Aut}_3(ID_i, \mathsf{dk}_i, C_i, ID_j, C_j)$: The algorithm takes as input two users' identities (ID_i, ID_j), the secret key dk_{ID_i} of user i, and two ciphertexts (C_i, C_j) encrypted under ID_i and ID_j, respectively, and outputs a Type-3 trapdoor $td_{(i,3,C_i,C_j)}$.
11) $\mathsf{Test}_3(C_i, td_{(i,3,C_i,C_j)}, C_j, td_{(j,3,C_j,C_i)})$: The algorithm takes as input two ciphertexts (C_i, C_j) and two Type-3 trapdoors $(td_{(i,3,C_i,C_j)}, td_{(j,3,C_j,C_i)})$, and outputs 1 if C_i and C_j contain the same message and 0 otherwise.

Type-4 Authorization:

12) $\mathsf{Aut}_4(ID_i, \mathsf{dk}_{ID_i}, C_i)$: Given a user's identity ID_i and and its secret key dk_{ID_i}, the algorithm outputs Type-4 trapdoors $td_{(i,4,C_i)}$ and $td_{(i,4)}$.
13) $\mathsf{Test}_4(C_i, td_{(i,4,C_i)}, C_j, td_{(j,4)})$: The algorithm takes as input two ciphertexts (C_i, C_j) and two Type-4 trapdoors $(td_{(i,4,C_i)}, td_{(j,4)})$, and outputs 1 if C_i and C_j contain the same message and 0 otherwise.

Remark: All the test algorithms should have the public information (e.g. master public key and identity) of i and j as input. Here we omit them for simplicity

IV. OUR IBEET-FA SCHEME

All the schemes make use of (symmetric) bilinear pairings. Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be cyclic groups with prime order p, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear pairing. Let ℓ_1, ℓ_2, n such that $\ell_1 \le \ell_2 \le n$ be three non-negative integers and A be a string of n bits. We denote by $[A]_{\ell_1}^{\ell_2}$ the substring of A which starts from the ℓ_1-th bit and ends at the ℓ_2-th bit.

Our construction of IBEET-FA combines the ideas of PKEET-FA [8] and IBEET [24] schemes. The secret key of each user in IBEET-FA has two parts, one for trapdoor generation and the other for decryption. Our IBEET-FA scheme works as below.

1) $\mathsf{Setup}(1^k)$: Given the security parameter k, the algorithm generates bilinear pairing parameters $\{\mathbb{G}_1, \mathbb{G}_T, p, g, \hat{e}\}$, and chooses three cryptographic hash functions, $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_T \times \mathbb{G}_1^3 \to \{0,1\}^{2\ell_1}$ and $H_3 : \mathbb{G}_T \times \mathbb{G}_1^3 \times \{0,1\}^* \to \{0,1\}^{2\ell_2}$, where ℓ_1 and ℓ_2 are the lengths of an element of $\mathbb{G}_1$ and $\mathbb{Z}_p$, respectively. It outputs $\mathcal{K} = \{\mathbb{G}_1, \mathbb{G}_T, p, g, \hat{e}, H_1, H_2, H_3\}$.
2) $\mathsf{KeyGen}(\mathcal{K})$: Randomly choose $s_1, s_2 \in \mathbb{Z}_p$ and compute $Y_1 = g^{s_1}$, $Y_2 = g^{s_2}$. Output the master secret key
$$\mathsf{msk} = s = (s_1, s_2)$$
and master public key
$$\mathsf{mpk} = Y = (Y_1, Y_2).$$
3) $\mathsf{Extract}(ID, s)$: Output the secret key dk_{ID} as follows:
$$\mathsf{dk}_{ID} = (\mathsf{dk}_{ID,1}, \mathsf{dk}_{ID,2}),$$
where $\mathsf{dk}_{ID,1} = H_1(ID)^{s_1}$ and $\mathsf{dk}_{ID,2} = H_1(ID)^{s_2}$.
4) $\mathsf{Encrypt}(M, ID, Y)$: Choose at random $r_1, r_2, r_3 \in \mathbb{Z}_p$, and compute the followings:
$$\begin{aligned} C_1 &= g^{r_1}, C_2 = g^{r_2}, C_4 = g^{r_3}, \\ C_3 &= (M^{r_1} \| H_1(ID)^{r_1 \cdot r_2}) \\ &\quad \oplus H_2(\hat{e}(H_1(ID), Y_1)^{r_3}, C_1, C_2, C_4), \\ C_5 &= (M \| r_1) \oplus H_3(\hat{e}(H_1(ID), Y_2)^{r_3}, C_1, C_2, C_3, C_4). \end{aligned}$$
Return the ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$.
5) $\mathsf{Decrypt}(C, ID, \mathsf{dk}_{ID}, Y)$: Compute
$$(M \| r_1) = C_5 \oplus H_3(\hat{e}(\mathsf{dk}_{ID,2}, C_4), C_1, C_2, C_3, C_4),$$
and
$$(A \| B) = C_3 \oplus H_2(\hat{e}(\mathsf{dk}_{ID,1}, C_2), C_1, C_2, C_4).$$
Output M if and only if all the following equations hold:
$$C_1 = g^{r_1}, \quad A = M^{r_1}, \quad \hat{e}(B, g) = \hat{e}(H_1(ID)^{r_1}, C_2).$$
6) $\mathsf{Aut}_1(ID_i, \mathsf{dk}_{ID_i})$: Output
$$td_{(i,1)} := \mathsf{dk}_{ID_i,1} = H_1(ID_i)^{s_1}.$$
7) $\mathsf{Test}_1(C_i, td_{(i,1)}, C_j, td_{(j,1)})$: Compute
$$\begin{aligned} M_i^{r_{i,1}} &= [C_{i,3} \oplus H_2(\hat{e}(td_{(i,1)}, C_{i,4}), \\ &\qquad C_{i,1}, C_{i,2}, C_{i,4})]_{\ell_1}^{2\ell_1-1}, \\ M_j^{r_{j,1}} &= [C_{j,3} \oplus H_2(\hat{e}(td_{(j,1)}, C_{j,4}), \\ &\qquad C_{j,1}, C_{j,2}, C_{j,4})]_{\ell_1}^{2\ell_1-1}, \end{aligned}$$
and output 1 if
$$\hat{e}(M_i^{r_{i,1}}, C_{j,1}) = \hat{e}(M_j^{r_{j,1}}, C_{i,1}) \tag{1}$$
holds, and 0 otherwise.
8) $\mathsf{Aut}_2(ID_i, \mathsf{dk}_{ID_i}, C_i)$: Output
$$\begin{aligned} td_{(i,2,C_i)} &= [H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, C_{i,4}), C_{i,1}, C_{i,2}, C_{i,4})]_{\ell_1}^{2\ell_1-1} \\ &= [H_2(\hat{e}(H_1(ID_i)^{s_1}, g^{r_{i,3}}), C_{i,1}, C_{i,2}, \\ &\qquad C_{i,4})]_{\ell_1}^{2\ell_1-1}. \end{aligned}$$
9) $\mathsf{Test}_2(C_i, td_{(i,2,C_i)}, C_j, td_{(j,2,C_j)})$: Compute
$$\begin{aligned} M_i^{r_{i,1}} &= [C_{i,3}]_{\ell_1}^{2\ell_1-1} \oplus td_{(i,2,C_i)}, \\ M_j^{r_{j,1}} &= [C_{j,3}]_{\ell_1}^{2\ell_1-1} \oplus td_{(j,2,C_j)}, \end{aligned}$$
and output 1 if Eq. (1) holds, and 0 otherwise.
10) $\mathsf{Aut}_3(ID_i, \mathsf{dk}_i, C_i, ID_j, C_j)$: Compute
$$\begin{aligned} &M_i^{r_{i,1}} \| H_1(ID_i)^{r_{i,1} \cdot r_{i,2}} \\ &\qquad = C_{i,3} \oplus H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, C_{i,4}), C_{i,1}, C_{i,2}, C_{i,4}), \end{aligned}$$

and compute

$$td_{(i,3,C_i,C_j)} = (td_{(i,3,C_i,C_j),1}, td_{(i,3,C_i,C_j),2}),$$

where

$$\begin{aligned} td_{(i,3,C_i,C_j),1} &= M_i^{r_{i,1}} \cdot H_1(ID_i)^{r_{i,1} \cdot r_{i,2}}, \\ td_{(i,3,C_i,C_j),2} &= \hat{e}(H_1(ID_i)^{r_{i,1} \cdot r_{i,2}}, C_{j,1}). \end{aligned}$$

Output $td_{(i,3,C_i,C_j)}$.

11) $\mathsf{Test}_3(C_i, td_{(i,3,C_i,C_j)}, C_j, td_{(j,3,C_j,C_i)})$: Output 1 if

$$\frac{\hat{e}(C_{i,1}, td_{(j,3,C_j,C_i),1})}{\hat{e}(C_{j,1}, td_{(i,3,C_i,C_j),1})} = \frac{td_{(j,3,C_j,C_i),2}}{td_{(i,3,C_i,C_j),2}}$$

holds, and 0 otherwise.

12) $\mathsf{Aut}_4(ID_i, \mathsf{dk}_{ID_i}, C_i)$: Output

$$\begin{aligned} td_{(i,4,C_i)} &= td_{(i,2,C_i)} \\ &= [H_2(\hat{e}(H_1(ID_i)^{s_1}, g^{r_{i,3}}), C_{i,1}, \\ &\quad C_{i,2}, {}_{i,4})]_{\ell_1}^{2\ell_1-1}, \\ td_{(i,4)} &= td_{(i,1)} = H_1(ID_i)^{s_1}. \end{aligned}$$

13) $\mathsf{Test}_4(C_i, td_{(i,4,C_i)}, C_j, td_{(j,4)})$: Compute

$$\begin{aligned} M_i^{r_{i,1}} &= [C_{i,3}]_{\ell_1}^{2\ell_1-1} \oplus td_{(i,4,C_i)}, \\ M_j^{r_{j,1}} &= [C_{j,3} \oplus H_2(\hat{e}(td_{(j,1)}, C_{j,4}), C_{j,1}C_{j,2}, \\ &\quad C_{j,4})]_{\ell_1}^{2\ell_1-1}, \end{aligned}$$

and output 1 if Eq. (1) holds, and 0 otherwise.

V. SECURITY ANALYSIS

In this section, we first define the security models of IBEET-FA, then, based on the security model, we prove our IBEET-FA scheme is OW-ID-CCA secure against Type-I adversaries and IND-ID-CCA secure against Type-II adversaries in the random oracle model.

A. SECURITY MODELS

Same as [8], we omit the security analysis of Type-4 authorization as it can be obtained from Type-1 and Type-2 authorizations. We take into account the following two types of adversaries.

- Type-I: The adversary $\mathcal{A}_1$ with authorization tries to retrieve the message M from the ciphertext.
- Type-II: The adversary $\mathcal{A}_2$ without authorization tries to distinguish to which message a given ciphertext is related.

Security against Type-I adversaries is the one-wayness under chosen-identity and chosen ciphertext attacks (OW-ID-CCA), and security against Type-II adversaries is the indistinguishability under chosen-identity and chosen ciphertext attacks (IND-ID-CCA). The two security properties are defined via the following games.

Game I: Let $\mathcal{A}_1$ be a Type-I adversary. Assume that the target receiver has index t $(1 \leq t \leq n)$. Consider the following game played between a challenger $\mathcal{C}_1$ and the adversary $\mathcal{A}_1$.

1) Setup: $\mathcal{C}_1$ generates the public parameter $\mathcal{K}$ and master public key mpk and sends them to $\mathcal{A}_1$.
2) Phase 1: $\mathcal{A}_1$ is allowed to issue the following queries for polynomially many times.
   - Extract queries: Given an identity ID_i, $\mathcal{C}_1$ computes and returns the corresponding secret key dk_{ID_i}.
   - Decryption queries: Given (ID_i, C_i), $\mathcal{C}_1$ decrypts C_i with respect to ID_i, and returns the result M_i (which might be $\perp$) to $\mathcal{A}_1$.
   - Authorization queries: For Type-α $(\alpha = 1, 2, 3)$ authorization:
     - given ID_i, $\mathcal{C}_1$ returns $td_{i,1}$;
     - given (ID_i, C_i), $\mathcal{C}_1$ returns $td_{i,2,C_i}$;
     - given (ID_i, C_i, ID_j, C_j), $\mathcal{C}_1$ returns $td_{i,3,C_i,C_j}$.
3) Challenge: $\mathcal{A}_1$ picks a target identity ID_t with the only constraint that ID_t did not appear in extract queries, and sends it to the challenger. $\mathcal{C}_1$ then randomly picks a message M, computes $C \leftarrow \mathsf{Encrypt}(M, ID_t, \mathsf{mpk})$, and returns C to the adversary.
4) Phase 2: $\mathcal{A}_1$ continues to issuing queries as in Phase 1, with constraints that C did not appear in decryption queries and ID_t did not appear in extract queries.
5) Guess: $\mathcal{A}_1$ returns a guessing message M' and wins the game if $M' = M$.

We denote $\mathcal{A}_1$'s advantage in Game I by

$$\mathrm{Adv}_{\mathrm{IBEET-FA},\mathcal{A}_1}^{\mathrm{OW-ID-CCA,Type-}\alpha}(k) = \Pr[M' = M].$$

Definition 2 (OW-ID-CCA): An IBEET-FA scheme is one-way under chosen identity and chosen ciphertext attacks (OW-ID-CCA secure) if for any PPT Type-I adversary $\mathcal{A}_1$, its advantage $\mathrm{Adv}_{\mathrm{IBEET-FA},\mathcal{A}_1}^{\mathrm{OW-ID-CCA,Type-}\alpha}(k)$ is negligible in the security parameter k.

Game II: Let $\mathcal{A}_2$ be a Type-II adversary. Assume that the target receiver has index t $(1 \leq t \leq n)$. Consider the following game played between a challenger $\mathcal{C}_2$ and the adversary $\mathcal{A}_2$.

1) Setup: $\mathcal{C}_2$ generates the public parameter $\mathcal{K}$ and master public key mpk, and sends them to $\mathcal{A}_2$.
2) Phase 1: $\mathcal{A}_2$ is allowed to issue queries for polynomially many times as in **Game I**.
3) Challenge: $\mathcal{A}_2$ submits two equal-length messages M_0, M_1 and a target identity ID_t, with the constraint that ID_t did not appear in extract queries nor Type-1 authorization queries. $\mathcal{C}_2$ then randomly chooses a bit $b \in \{0, 1\}$ and computes $C^* \leftarrow \mathsf{Encrypt}(M_b, ID_t, \mathsf{mpk})$. It returns C^* to the adversary.
4) Phase 2: $\mathcal{A}_2$ continues to issuing queries as in Phase 1, with the following constraints:
   - (ID_t, C^*) did not appear in decryption queries,
   - ID_t did not appear in extract queries nor Type-1 authorization queries, and (ID_t, C^*) did not appear in the authorization queries.
5) Guess: Finally, $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

We denote $\mathcal{A}_2$'s advantage in Game II by

$$\mathrm{Adv}^{\mathrm{IND-ID-CCA,Type-}\alpha}_{\mathrm{IBEET-FA},\mathcal{A}_2}(k) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

Definition 3 (IND-ID-CCA): An IBEET-FA scheme is indistinguishable under chosen identity and chosen ciphertext attacks (IND-ID-CCA secure) if for any PPT Type-II adversary $\mathcal{A}_2$, its advantage $\mathrm{Adv}^{\mathrm{IND-ID-CCA,Type-}\alpha}_{\mathrm{IBEET-FA},\mathcal{A}_2}(k)$ is negligible in the security parameter k.

Remark: If an IBEET-FA scheme is IND-ID-CCA secure, no PPT adversary can generate a valid Type-3 trapdoor of an unauthorized ciphertext with non-negligible probability. Suppose that $\mathcal{A}'$ is an adversary who can generate a valid Type-3 trapdoor of an unauthorized ciphertext. We can use it to build another algorithm $\mathcal{A}_2$ to break IND-ID-CCA security with non-negligible advantage as follows:

1) $\mathcal{A}_2$ randomly chooses an identity ID' and issues an authorization query to obtain Type-1 trapdoor of ID'.
2) $\mathcal{A}_2$ produces a ciphertext C' of M_0 under ID'.
3) Given the challenge ciphertext C^*, $\mathcal{A}_2$ runs $\mathcal{A}'$ to generate a Type-3 trapdoor with respect to C^* and C'. Then it runs the test algorithm to check whether C^* and C' contain the same message, and determines the bit b'.

Similarly, there is no PPT adversary which is able to produce a valid Type-2 or Type-1 trapdoor of an unauthorized ciphertext with non-negligible probability.

B. OW-ID-CCA SECURITY

Theorem 1 (OW-ID-CCA): Our IBEET-FA scheme above is OW-ID-CCA secure against Type-I adversaries in the random oracle model if BDH assumption holds.

Proof: Let $\mathcal{A}$ be a Type-I adversary against the OW-ID-CCA security of our IBEET-FA scheme, and C be the challenger. Consider the following games. Below $\mathbf{S}_i$ $(i = 1, 2, 3)$ denotes the the event that the adversary wins in Game i.

Game 1: This is the original Type-I game.

1) Setup: In this phase, the challenger C generates the global parameter $\mathcal{K}$, the master public key Y, and the master secret key s as follows:

$$\mathcal{K} = \{\mathbb{G}_1, \mathbb{G}_T, p, g, \hat{e}\}, Y = (Y_1, Y_2) = (g^{s_1}, g^{s_2}) \in \mathbb{G}_1^2,$$
$$s = (s_1, s_2) \in \mathbb{Z}_p^2.$$

The challenger makes $\mathcal{K}$ and Y public.

2) Phase 1: The adversary $\mathcal{A}$ is allowed to issue queries to the following oracles.
   - Hash Oracles: We assume the adversary $\mathcal{A}$ can issue at most $q_{H_1}, q_{H_2}, q_{H_3}$ queries to hash oracles $\mathcal{O}_{H_1}, \mathcal{O}_{H_2}, \mathcal{O}_{H_3}$, respectively, and it does not repeat any query to the same oracle.
     - $\mathcal{O}_{H_1}$: The oracle maintains a list $L_{H_1} = \{(ID, h, H_1(ID))\}$ which is initially empty. Given an identity ID_i, it randomly chooses $h_i \leftarrow \mathbb{Z}_p$, returns $H_1(ID_i) = g^{h_i}$ to $\mathcal{A}$, and stores the tuple (ID_i, h_i, g^{h_i}) into L_{H_1}.
     - $\mathcal{O}_{H_2}$: The oracle maintains a list $L_{H_2} = \{(T, H_2(T))\}$ which is initially empty. Given an input T_i, it randomly selects a string $R \in \{0,1\}^{2\ell_1}$, returns $H_2(T_i) = R$ to $\mathcal{A}$, and stores (T_i, R) in L_{H_2}.
     - $\mathcal{O}_{H_3}$: The oracle maintains a list $L_{H_3} = \{(T, H_3(T))\}$ which is initially empty. Given an input T_i, it randomly selects a string $R \in \{0,1\}^{2\ell_2}$, returns $H_3(T_i) = R$ to $\mathcal{A}$, and stores (T_i, R) in L_{H_3}.
   - Extract Oracle: Given an identity ID_i, the oracle runs the hash oracle $\mathcal{O}_{H_1}$ to get the hash value $H_1(ID_i)$ and runs the $\mathsf{Extract}(ID_i, s)$ algorithm to generate the decryption key

$$\begin{aligned}\mathsf{dk}_{ID_i} &= (\mathsf{dk}_{ID_i,1}, \mathsf{dk}_{ID_i,2})\\ &= (H_1(ID_i)^{s_1}, H_1(ID_i)^{s_2}).\end{aligned}$$

   It returns dk_{ID_i} to $\mathcal{A}$.
   - Decrypt Oracle: Given an identity ID_i and $C = (C_1, C_2, C_3, C_4, C_5)$, the oracle $\mathcal{O}_D$ runs the extract oracle $\mathcal{O}_E$ to get the decryption key dk_{ID_i} and runs the $\mathsf{Decrypt}(C, ID_i, \mathsf{dk}_{ID_i}, Y)$ to get a message M (which might be $\perp$ if the decryption fails). It returns M to $\mathcal{A}$.
   - Trapdoor Oracle: Given a trapdoor query $(ID_i, \cdots)$, the oracle $\mathcal{O}_T$ runs $\mathsf{Extract}(ID_i, s)$ to get the secret key dk_{ID_i} and runs $\mathsf{Aut}_1(ID_i, \mathsf{dk}_{ID_i})$ to generate the Type-1 trapdoor $td_{(i,1)} = \mathsf{dk}_{ID_i,1} = H_1(ID_i)^{s_1}$ of ID_i. With the Type-1 trapdoor, it can compute the trapdoor of any other type. The oracle returns the corresponding trapdoor.
3) Challenge: At some time, the adversary $\mathcal{A}$ submits a challenge identity ID_t, which did not appear in the extract queries. Then, the challenger $\mathcal{C}$ randomly selects a message M and runs $\mathsf{Encrypt}(M, ID_t, Y)$ to generate a ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as below, and returns C^* to the adversary:

$$\begin{aligned}C_1^* &= g^{r_1}, \quad C_2^* = g^{r_2}, \quad C_4^* = g^{r_3},\\ C_3^* &= (M^{r_1} \| H_1(ID_t)^{r_1 \cdot r_2})\\ &\quad \oplus H_2(\hat{e}(H_1(ID_t), Y_1)^{r_3}, C_1^*, C_2^*, C_4^*),\\ C_5^* &= (M \| r_1) \oplus H_3(\hat{e}(H_1(ID_t), Y_2)^{r_3}, C_1^*, C_2^*, C_3^*, C_4^*).\end{aligned}$$

4) Phase 2: $\mathcal{A}$ continues to issuing queries as in phase 1, with constraints that ID_t cannot be submitted to $\mathcal{O}_E$ and (ID_t, C^*) cannot be submitted to $\mathcal{O}_D$.
5) Guess: The adversary $\mathcal{A}$ outputs a plaintext M' and wins if and only if $M' = M$. We have the following:

$$\Pr[\mathbf{S}_1] = \Pr[M' = M].$$

Game 2: It is the same as Game 1, except that the challenge ciphertext C^* is computed as follows:

$$\begin{aligned}C_1^* &= g^{r_1}, \quad C_2^* = g^{r_2}, \quad C_4^* = g^{r_3},\\ C_3^* &= (M^{r_1} \| H_1(ID_t)^{r_1 \cdot r_2})\\ &\quad \oplus H_2(\hat{e}(H_1(ID_t), Y_1)^{r_3}, C_1^*, C_2^*, C_4^*),\\ C_5^* &= (M \| r_1) \oplus W_1.\end{aligned}$$

The only difference is that we now use a random W_1 to replace $H_3(\hat{e}(H_1(ID), Y_2)^{r_3}, C_1^*, C_2^*, C_3^*, C_4^*)$. Notice that in Game 1, $H_3(\hat{e}(H_1(ID_t), Y_2)^{r_3}, \cdots)$ in the challenge ciphertext is random unless the adversary queried the hash input $(\hat{e}(H_1(ID_t), Y_2)^{r_3}, \cdots)$. Denote by $\mathbf{E}_1$ the event that the adversary $\mathcal{A}$ queried $(\hat{e}(H_1(ID_t), Y_2)^{r_3}, \cdots)$ to $\mathcal{O}_{H_3}$. If $\mathbf{E}_1$ does not happen, Game 2 is identical to Game 1 in $\mathcal{A}$'s view. We have that

$$\Pr[\mathbf{S}_2] = \Pr[\mathbf{S}_1 | \overline{\mathbf{E}_1}]. \tag{2}$$

Therefore, we have that

$$|\Pr[\mathbf{S}_1] - \Pr[\mathbf{S}_2]| \leq \Pr[\mathbf{E}_1]. \tag{3}$$

Lemma 1: $\Pr[\mathbf{E}_1]$ is negligible if the BDH assumption holds.

The proof is given later (on page 25416).

Game 3: It is the same as Game 2 except that C_5^* in the challenge ciphertext is replaced with a random string of length $2\ell_2$.

Notice that in Game 2, C_5^* is the exclusive OR of $M\|r_1$ with a random string, and thus it is also random. Therefore, the adversary's view in Game 3 is identical to that in Game 2, and we have that

$$\Pr[\mathbf{S}_3] = \Pr[\mathbf{S}_2]. \tag{4}$$

Lemma 2: $\Pr[\mathbf{S}_3]$ is negligible if BDH assumption holds.

The proof is given later (on page 25417).

Combining Lemma 1 and 2 and Eqs. (3) and (4), we have that

$$\Pr[\mathbf{S}_1] \leq \Pr[\mathbf{S}_2] + \Pr[\mathbf{E}_1] = \Pr[\mathbf{S}_3] + \Pr[\mathbf{E}_1] \leq \mathrm{negl}(k).$$

This completes the proof of Theorem 1. ■

Proof of Lemma 1: Assuming that event $\mathbf{E_1}$ happens with probability ϵ_{E_1} in Game 1 (page 25415). We build an algorithm $\mathcal{B}$ to solve the BDH problem.

Given $(\mathbb{G}_1, \mathbb{G}_T, p, \hat{e}, g, g_1 = g^{x_1}, g_2 = g^{x_2}, g_3 = g^{x_3})$, where x_1, x_2, x_3 are random elements of $\mathbb{Z}_p$, algorithm $\mathcal{B}$ randomly chooses $s_1 \leftarrow \mathbb{Z}_p$, and sets

$$\mathsf{mpk} = Y = (Y_1, Y_2) = (g^{s_1}, g_1),$$

and keeps s_1 secretly. Notice that the second component of msk is $s_2 = x_1$, which is unknown to $\mathcal{B}$. It randomly chooses $t^* \leftarrow \{1, \cdots, p\}$ as its guess of the index of the challenge identity chosen by the adversary. $\mathcal{B}$ then gives the system parameter and mpk to the adversary, and answers its queries as below.

1) Phase 1: The adversary $\mathcal{A}$ is allowed to issue queries to the following oracles.
   - $\mathcal{O}_{H_1}$: If the input identity ID_i is the t^*-th query, $\mathcal{B}$ sets $h_i = \perp$ and returns $H_1(ID_i) = g_2$. Otherwise, it randomly chooses $h_i \leftarrow \mathbb{Z}_p$ and returns $H_1(ID_i) = g^{h_i}$. In either case, $\mathcal{B}$ stores $(ID_i, h_i, H_1(ID_i))$ into the list L_{H_1} (which is initially empty).
   - $\mathcal{O}_{H_2}$: Same with that in Game 2 (page 25415).
   - $\mathcal{O}_{H_3}$: Same with that in Game 2 (page 25415).
   - Extract Oracle $\mathcal{O}_E$: Given an identity ID_i, if it is the t^*-th query to $\mathcal{O}_{H_1}$, $\mathcal{B}$ aborts the game. Otherwise, it retrieves the tuple $(ID_i, h_i, H_1(ID_i))$ from L_{H_1}, and computes the secret key as below:

   $$\mathsf{dk}_{ID_i} = (\mathsf{dk}_{ID_i,1}, \mathsf{dk}_{ID_i,2}) = (Y_1^{h_i}, Y_2^{h_i}).$$

   $\mathcal{B}$ returns dk_{ID_i} to the adversary.
   - Decrypt Oracle $\mathcal{O}_D$: Given an identity ID_i and a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$, if ID_i is not the t^*-th query to $\mathcal{O}_{H_1}$, $\mathcal{B}$ runs the Extract Oracle $\mathcal{O}_E$ to obtain the corresponding secret key dk_{ID_i}, and returns the decryption result $\mathsf{Decrypt}(C, ID_i, \mathsf{dk}_{ID_i}, Y)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ traverses the list L_{H_3} and computes $(M\|r_1) = C_5 \oplus H_3(T)$ for each tuple $(T, H_3(T))$ with $T = (T_1, C_1, C_2, C_3, C_4)$. It stops traversing if all equations below hold:

   $$C_1 = g^{r_1}, \quad A = M^{r_1},$$
   $$\hat{e}(B, g) = \hat{e}(H_1(ID_i)^{r_1}, C_2),$$

   where $(A\|B) = C_3 \oplus H_2(\hat{e}(H_1(ID_i)^{s_1}, C_2), \cdots)$. If there is no such an M, $\mathcal{B}$ returns $\perp$.
   - Trapdoor Oracle $\mathcal{O}_T$: Same with that in Game 2 (page 25415).
2) Challenge: At some point, the adversary $\mathcal{A}$ submits a challenge identity ID_t. If ID_t is not the t^*-th query to $\mathcal{O}_{H_1}$, $\mathcal{B}$ aborts the game; otherwise, $\mathcal{B}$ randomly selects a message M_t and computes the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as below, and returns C^* to the adversary:

   $$C_1^* = g_2^{r_1}, \quad C_2^* = g^{r_2}, \quad C_4^* = g_3,$$
   $$C_3^* = (M_t^{r_1} \| g_2^{r_1 \cdot r_2}) \oplus H_2(\hat{e}(g_2, C_4^*)^{s_1}, C_1^*, C_2^*, C_4^*),$$
   $$C_5^* = (M_t \| r_1) \oplus W_1,$$

   where $r_1, r_2, r_3 \in Z_p^3$ are random.
3) Phase 2: The adversary continues to issuing queries as in Phase 1. The constraints are that the challenge identity ID_t did not appear in Extract queries, and (ID_t, C^*) did not appear in Decrypt queries.
4) Guess: Finally, $\mathcal{A}$ outputs a plaintext M_t', and wins the game if $M_t' = M_t$.

Notice that if $\mathcal{B}$'s guess of t^* is correct, i.e. ID_{t^*} is chosen by the adversary as the challenge identity, the view of the adversary is identical to that in a real attack. Denote by abt the event that $\mathcal{B}$ aborts. Due to the random choice of t^*, we have that $\Pr[\overline{\mathsf{abt}}] \geq 1/q_{H_1}$.

Conditioned on that $\mathcal{B}$ does not abort, if $\mathcal{A}$ did not query $\mathcal{O}_{H_3}$ with input $(\hat{e}(H_1(ID_t), Y_2)^{x_3}, \cdots) = (\hat{e}(g, g)^{x_1\ x_2\ x_3}, \cdots)$, the corresponding hash value is random to $\mathcal{A}$. Besides, C_5^* is also random. Therefore, the challenge ciphertext hides the message M_t perfectly, and the probability that the adversary finds the correct M_t is negligible. Hence, with overwhelming probability $\mathcal{A}$ queried $\mathcal{O}_{H_3}$ with input

$\hat{e}(g,g)^{x_1 x_2 x_3}$. $\mathcal{B}$ randomly picks a tuple $(T, H_3(T))$ from the list L_{H_3}, where $T = (T_1, \cdots, T_5)$, and outputs T_1. The probability that the output of $\mathcal{B}$ is the solution to the given BDH problem instance is at least $\epsilon_{E_1}/q_{H_1}q_{H_3} - \text{negl}(k)$. If ϵ_{E_1} is non-negligible, so is $\mathcal{B}$'s advantage in breaking the BDH assumption. Therefore, the event $\mathbf{E_1}$ happens with negligible probability.

This completes the proof of Lemma 1. ■

Proof of Lemma 2: If $\Pr[\mathbf{S_3}]$ in Game 3 (page 25416) is non-negligible, we can build an algorithm $\mathcal{B}$ to solve the BDH problem. Given $(\mathbb{G}_1, \mathbb{G}_T, p, \hat{e}, g, g_1 = g^{x_1}, g_2 = g^{x_2}, g_3 = g^{x_3})$, where x_1, x_2, x_3 are random elements of $\mathbb{Z}_p$, algorithm $\mathcal{B}$ randomly chooses a number $s_2 \leftarrow \mathbb{Z}_p$, and sets

$$\mathsf{mpk} = Y = (Y_1, Y_2) = (g_1, g^{s_2}),$$

and keeps s_2 secretly. Notice that the first component of msk is $s_1 = x_1$, which is unknown to $\mathcal{B}$. It randomly chooses $t^* \leftarrow \{1, \cdots, p\}$ as its guess of the index of the challenge identity chosen by the adversary. $\mathcal{B}$ then gives the system parameter and mpk to the adversary, and answers its queries as below.

1) Phase 1: Same as that in the proof of Lemma 1.
2) Challenge: At some point, the adversary $\mathcal{A}$ submits a challenge identity ID_t. If ID_t is not the t^*-th query to $\mathcal{O}_{H_1}$, $\mathcal{B}$ aborts the game; otherwise, $\mathcal{B}$ randomly selects a message M_t and computes the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as below, and returns C^* to the adversary:
$$C_1^* = g_2^{r_1}, \quad C_2^* = g^{r_2}, \quad C_4^* = g_3,$$
$$C_3^* = (M_t^{r_1} \| g_2^{r_1 \cdot r_2}) \oplus H_2(Z, C_1^*, C_2^*, C_4^*),$$
$$C_5^* = W_2,$$
where $r_1, r_2, r_3 \in \mathbb{Z}_p^3$ and $Z \in \mathbb{G}_T$ are random.
3) Phase 2: The adversary continues to issuing queries as in Phase 1. The constraints are that the challenge identity ID_t did not appear in Extract queries, and (ID_t, C^*) did not appear in Decrypt queries.
4) Guess: $\mathcal{A}$ outputs a message M_t', and wins the game if $M_t' = M_t$.

Denote by abt the event $\mathcal{B}$ aborts the game. Notice that if $\mathcal{B}$ does not abort the game, the view of the adversary is identical to a real attack. Same as that in the proof of Lemma 1 we have $\Pr[\overline{\mathsf{abt}}] \geq 1/q_{H_1}$.

Suppose that $\mathcal{B}$ does not abort. If $\mathcal{A}$ did not query $\mathcal{O}_{H_2}$ with input $(\hat{e}(H_1(ID_t), Y_1)^{x_3}, \cdots) = (\hat{e}(g,g)^{x_1 x_2 x_3}, \cdots)$, the corresponding hash value is totally random to $\mathcal{A}$. Therefore the challenge ciphertext hides the message M_t perfectly, and $\mathcal{A}$ has negligible probability to find M_t. Thus, with overwhelming probability $\mathcal{A}$ issues a query to $\mathcal{O}_{H_2}$ on input $(\hat{e}(g,g)^{x_1 x_2 x_3}, \cdots)$, $\mathcal{B}$ randomly picks a tuple $(T, H_2(T))$ from the list L_{H_2}, where $T = (T_1, \cdots, T_4)$. The probability that T_1 is the solution of the given BDH instance will be at least $\Pr[\mathbf{S_3}]/q_{H_1}q_{H_2}$. Hence $\Pr[\mathbf{S_3}]$ is negligible if the BDH assumption holds.

This completes the proof of Lemma 2. ■

C. IND-ID-CCA SECURITY

Theorem 2 (IND-ID-CCA): Our proposed IBEET-FA scheme is IND-ID-CCA secure in the random oracle model if DBDH assumption holds.

Proof: Assume that $\mathcal{A}_2$ is a Type-II adversary who breaks the IND-ID-CCA security of our proposed IBEET-FA scheme with advantage ϵ_2. Then we construct an algorithm $\mathcal{B}$ to solve the DBDH problem. Let $(\mathbb{G}_1, \mathbb{G}_T, p, \hat{e}, g)$ be bilinear pairing parameters as described in the scheme, and let $\mathcal{C}_{DBDH}$ be the challenger of DBDH problem.

Given $(g, g^{x_1}, g^{x_2}, g^{x_3}, Z)$ where Z is either equal to $\hat{e}(g,g)^{x_1 x_2 x_3}$ (i.e. $b = 0$) or a random element of $\mathbb{G}_T$ (i.e. $b = 1$), $\mathcal{B}$ randomly selects $s_1 \leftarrow \mathbb{Z}_p$ and sets the master public key $\mathsf{mpk} = Y = (Y_1, Y_2) = (g^{x_1 \cdot s_1}, g^{x_1})$. Suppose that the adversary $\mathcal{A}_2$ issues at most $q_{H_1}, q_{H_2}, q_{H_3}$ queries to hash oracles $\mathcal{O}_{H_1}, \mathcal{O}_{H_2}, \mathcal{O}_{H_3}$, respectively. $\mathcal{B}$ chooses at random $t^* \leftarrow \{1, \cdots, q_{H_1}\}$ (as a guess of which identity would be chosen as the challenge identity by the adversary), and gives the system public parameters and mpk to the adversary $\mathcal{A}_2$, and answers $\mathcal{A}_2$'s queries as below.

1) Phase 1: $\mathcal{A}_2$ is allowed to issue the following oracles. Again, we assume that the adversary does not repeat a query to the same oracle.
   - Hash Oracles:
     - $\mathcal{O}_{H_1}$: Given the i-th distinct query ID_i, if $i = t^*$, $\mathcal{B}$ sets $h_i = \perp$ and $H_1(ID_{t^*}) = g^{x_2}$; otherwise, it randomly selects $h_i \leftarrow \mathbb{Z}_p$ and sets $H_1(T_i) = g^{h_i}$. In either case, $\mathcal{B}$ returns $H_1(ID_i)$ to the adversary, and stores $(ID_i, h_i, H_1(ID_i))$ into a list L_{H_1} which is initially empty.
     - $\mathcal{O}_{H_2}$: Same as that in the proof of Theorem 1.
     - $\mathcal{O}_{H_3}$: Same as that in the proof of Theorem 1.
   - Extract Oracle: Given an identity ID_i, $\mathcal{B}$ retrieves the corresponding tuple $(ID_i, h_i, H_1(ID_i))$ from L_{H_1}. If $h_i = \perp$, which means that ID_i is the t^*-th query to $\mathcal{O}_{H_1}$, $\mathcal{B}$ aborts and outputs a random bit b'. Otherwise, $\mathcal{B}$ computes the secret key
   $$\mathsf{dk}_{ID_i} = (\mathsf{dk}_{ID_i,1}, \mathsf{dk}_{ID_i,2}) = (Y_1^{h_i}, Y_2^{h_i}),$$
   and returns dk_{ID_i}.
   - Decryption Oracle: Given an identity ID_i and a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$, $\mathcal{B}$ runs the following algorithm and gets the *result* $= M \| T'$ or $\perp$. $\mathcal{B}$ returns M or $\perp$. T' will be used by the trapdoor oracle.
   - Trapdoor Oracle: Given the identity ID_i, $\mathcal{B}$ runs the hash oracle $\mathcal{O}_{H_1}$ to get the value $H_1(ID_i)$ and returns the trapdoor according to the following cases.
     - Type-1: Given ID_i, if $ID_i = ID_{t^*}$, $\mathcal{B}$ aborts and outputs a random bit b'. Otherwise, $\mathcal{B}$ returns $(ID_i, h_i, H_1(ID_i))$ from L_{H_1}, and computes
     $$td_{i,1} = \mathsf{dk}_{ID_i,1} = Y_1^{h_i}.$$

Algorithm 1

```
1:  $result = \perp$
2:  for each $(T, H_3(T)) \in L_{H_3}$ do
3:      parse $T$ as $T = (T_1, T_2, T_3, T_4)$
4:      compute $(M||r_1) = C_5 \oplus H_3(T)$
5:      if $C_1 = M^{r_1} \wedge T_2 = C_1 \wedge T_3 = C_2 \wedge T_4 = C_4$ then
6:          for each $(T', H_2(T')) \in L_{H_2}$ do
7:              parse $T'$ as $T' = (T'_1, T'_2, T'_3, T'_4, T'_5)$
8:              compute $A = [C_3 \oplus H_2(T')]_{l_1}^{2l_1-1}$
9:              if $A = M^{r_1} \wedge T'_2 = C_1 \wedge T'_3 = C_2 \wedge T'_4 = C_3 \wedge T'_5 = C_4$ then
10:                 $result = M\|T'$
11:                 return $result$
12:             end if
13:         end for
14:     end if
15: end for
16: return $result$
```

- Type-2: Given (ID_i, C_i), if $ID_i \neq ID_{t^*}$, return

$$\begin{aligned} td_{i,2} &= H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, C_{i,2}), C_{i,1}, C_{i,2}, C_{i,4}) \\ &= H_2(\hat{e}(Y_1^{h_i}, g^{r_{i,2}}), \cdots). \end{aligned}$$

  Otherwise, $\mathcal{B}$ runs the decryption oracle $\mathcal{O}_D$ to get the value *result*. If $result = \perp$, return $\perp$, otherwise, return $td_{i,2} = H_2(T')$
- Type-3: Given $(ID_i, C_i), (ID_j, C_j)$, if $ID_i \neq ID_{t^*}$, return $td_{i,3} = (td_{i,3,1}, td_{i,3,2})$ where

$$\begin{aligned} td_{i,3,1} &= [C_{i,3} \oplus H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, C_{i,4}), \cdots)]_0^{\ell_1-1} \\ &\quad \cdot [C_{i,3} \oplus H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, C_{i,4}), \cdots)]_{\ell_1}^{2\ell_1-1} \\ &= M_i^{r_{i,1}} \cdot H_1(ID_i)^{r_{i,1}\cdot r_{i,2}}, \\ td_{i,3,2} &= \hat{e}([C_{i,3} \oplus H_2(\hat{e}(\mathsf{dk}_{ID_i,1}, \\ &\quad C_{i,4}), \cdots)]_0^{\ell_1-1}, C_{j,1}) \\ &= \hat{e}(H_1(ID_i)^{r_{i,1}\cdot r_{i,2}}, C_{j,1}). \end{aligned}$$

  Otherwise, $\mathcal{B}$ runs the Type-2 trapdoor oracle with input (ID_i, C_i). Denote the output by $td_{i,2}$. Finally, $\mathcal{B}$ returns $td_{i,3} = (td_{i,3,1}, td_{i,3,2})$ where

$$\begin{aligned} td_{i,3,1} &= [C_{i,3} \oplus td_{i,2}]_0^{\ell_1-1} \cdot [C_{i,3} \oplus td_{i,2}]_{\ell_1}^{2\ell_1-1}, \\ td_{i,3,2} &= \hat{e}([C_{i,3} \oplus td_{i,2}]_0^{\ell_1-1}, C_{j,1}). \end{aligned}$$

Notice that every query to H_2 mentioned above should include $C_{i,1}, C_{i,2}, C_{i,4}$. We omit them here for simplicity.

2) Challenge: The adversary submits a challenge identity ID_t and two message M_0, M_1. If $ID_t \neq ID_{t^*}$, $\mathcal{B}$ aborts and outputs a random bit b'. Otherwise, $\mathcal{B}$ tosses a coin $\hat{b} \in \{0, 1\}$ and encrypts $M_{\hat{b}}$ by computing

$$\begin{aligned} &C_1^* = g^{r_1}, \quad C_2^* = g^{r_2}, \quad C_4^* = g^{x_3}, \\ &C_3^* = (M_{\hat{b}}^{r_1} \| H_1(ID_t)^{r_1\cdot r_2}) \oplus H_2(Z^{s_1}, C_1^*, C_2^*, C_4^*), \\ &C_5^* = (M_{\hat{b}} \| r_1) \oplus H_3(Z, C_1^*, C_2^*, C_3^*, C_4^*). \end{aligned}$$

$\mathcal{B}$ returns the ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ to the adversary.

3) Phase 2: $\mathcal{A}$ continues to issuing queries as in Phase 1 with the following constraints:
   a) (ID_t, C^*) did not appear in decryption queries;
   b) ID_t did not appear in extract queries nor Type-1 trapdoor queries, and (ID_t, C^*) did not appear in Type-2,3,4 trapdoor queries.
4) Guess: $\mathcal{A}$ outputs a bit $\hat{b}'$. If $\hat{b}' = \hat{b}$, $\mathcal{B}$ outputs a bit $b' = 0$; otherwise, it outputs $b' = 1$.

Notice that in the game above, $\mathcal{B}$ aborts and outputs a random bit if its guess of the challenge identity is wrong, i.e. $ID_{t^*} \neq ID_t$. Denote this event by abt, and we have that $\Pr[\overline{\mathsf{abt}}] \geq 1/q_{H_1}$. Conditioned on that abt does not occur, if $Z = \hat{e}(g, g)^{x_1\ x_2\ x_3}$, i.e. $b = 0$, the view of $\mathcal{A}_2$ is identical to that in a real attack, and $\mathcal{A}_2$ wins the game with advantage ϵ_2. If Z is a random element of $\mathbb{G}_T$, i.e. $b = 1$, all components of the challenge ciphertext are random and reveal nothing about the bit $\hat{b}$, thus the adversary wins the game only with probability at most 1/2. So we have that

$$\begin{aligned} &\Pr[b' = b|\overline{\mathsf{abt}}] \\ &\quad = \Pr[b' = 0 \wedge b = 0|\overline{\mathsf{abt}}] + \Pr[b' = 1 \wedge b = 1|\overline{\mathsf{abt}}] \\ &\quad = \frac{1}{2} + \frac{\epsilon_2}{2}. \end{aligned}$$

Therefore, we have that

$$\begin{aligned} \Pr[b' = b] &= \Pr[b' = b \wedge \mathsf{abt}] + \Pr[b' = b \wedge \overline{\mathsf{abt}}] \\ &\geq \frac{1}{2} + \frac{\epsilon_2}{2} \cdot \frac{1}{q_{H_1}}. \end{aligned}$$

If ϵ_2 is non-negligible, so is $\left|\Pr[b' = b] - \frac{1}{2}\right|$. This completes the proof of Theorem 2. ■

VI. EFFICIENCY ANALYSIS AND COMPARISON

In this section we analyze the efficiency of our IBEET-FA scheme, and compare the scheme with some related works, e.g. the PKEET-FA scheme [8], VPKEET [22] and the IBEET scheme [24], in terms of security, communicational complexity and computational complexity. Table 1 shows that our scheme has a comparable communicational complexity and the same level of security with PKEET-FA and IBEET. Table 2 shows that the computational costs of our scheme are comparable with PKEET-FA and IBEET schemes. In the tables we use $|\mathbb{G}_1|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ to denote the bit length of an element in $\mathbb{G}_1$, $\mathbb{G}_T$ and $\mathbb{Z}_p$, respectively, and use Pairing and Exp to denote the computational cost of evaluating a bilinear pairing and a modular exponentiation, respectively.

Due to the functional similarity, we implemented our scheme and PKEET-FA scheme in order to do a detailed comparison. The experiment platform is a VMware [32] virtual machine (VMware Workstation 12 Pro v12.1.1), with a two-core CPU and 4 GB memory and running Ubuntu 16.04 32-bit. The host machine has a quad-core 3.40GHz Intel i7-6700 CPU and 8 GB memory, and runs Windows 7 professional. We used the Type-A pairing in PBC library [33]

TABLE 1. Comparison of communicational complexity and security.

	$\|pk\|$	$\|sk\|$	$\|C\|$	Sec/w	Sec/o
PKEET-FA [8]	$3\|\mathbb{G}_1\|$	$3\|\mathbb{Z}_p\|$	$5\|\mathbb{G}_1\| + \|\mathbb{Z}_p\|$	OW-CCA	IND-CCA
VPKEET(1) [22]	$2\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_p\|$	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_p\|$	OW-CCA	IND-CCA
VPKEET(2) [22]	$2\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_p\|$	$4\|\mathbb{G}_1\| + \|\mathbb{Z}_p\|$	OW-CCA	IND-CCA
IBEET [24]	$1\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$5\|\mathbb{G}_1\| + \|\mathbb{Z}_p\|$	OW-ID-CCA	—
Our IBEET-FA	$1\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$5\|\mathbb{G}_1\| + \|\mathbb{G}_T\| + 2\|\mathbb{Z}_p\|$	OW-ID-CCA	IND-ID-CCA

Sec/w: Security against adversaries with trapdoors.
Sec/o: Security against adversaries without trapdoors.

TABLE 2. Comparison of computational complexity.

	Encrypt	Decrypt	Aut	Test	ID-based
PKEET-FA [8] Type-1	6Exp	5Exp	0	2Pairing + 2Exp	NO
PKEET-FA [8] Type-2	6Exp	5Exp	2Exp	2Pairing + 2Exp	NO
PKEET-FA [8] Type-3	6Exp	5Exp	2Pairing + 2Exp	2Pairing + 2Exp	NO
VPKEET(1) [22]	4Exp	2Exp	6Exp	4Pairing	NO
VPKEET(2) [22]	4Exp	2Exp	6Exp	5Pairing	NO
IBEET [24]	6Exp	2Pairing + 2Exp	-	4Pairing	YES
Our Type-1	7Exp	3Pairing + 2Exp	0	4Pairing	YES
Our Type-2	7Exp	3Pairing + 2Exp	Pairing	2Pairing	YES
Our Type-3	7Exp	3Pairing + 2Exp	2Pairing	2Pairing	YES

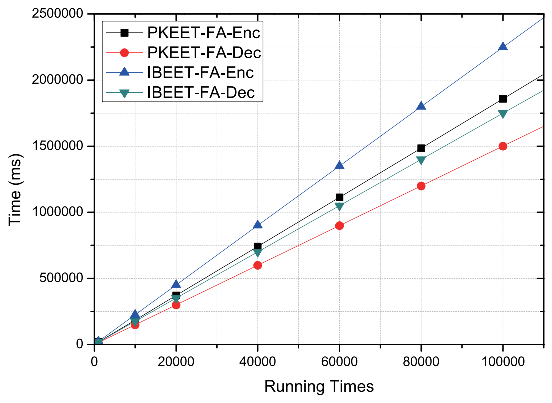


FIGURE 3. Efficiency comparison of encryption and decryption with PKEET-FA [8].

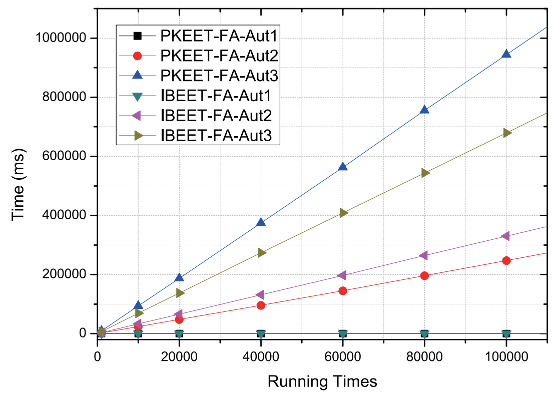


FIGURE 4. Efficiency comparison of authorization with PKEET-FA [8].

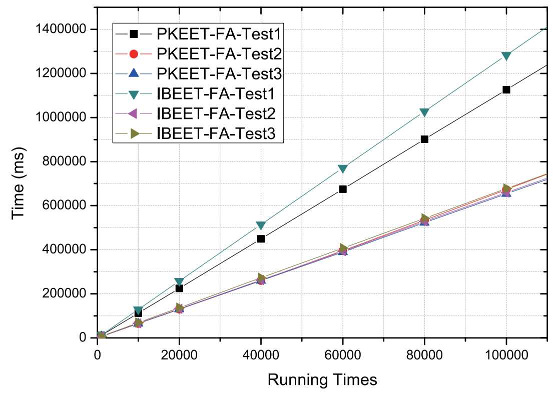


FIGURE 5. Efficiency comparison of test with PKEET-FA [8].

to implement the schemes. The elliptic curve $y^2 = x^3 + x$ over the field F_p with prime $p \equiv 3 \bmod 4$ [33] was used in the experiment, and the prime p is

$$
\begin{aligned}
p = {} & A7A73868E95FBA886EDEF8CE96E7217E364BB \\
& 946F5ED839628D1F80010940622A7AFDAF9B049 \\
& 744A459E54DAB7BA5BE92539E8FF9B4F30A3C \\
& F6230C28E284D97.
\end{aligned}
$$

Figure 3 shows that the computational costs of Encrypt and Decrypt algorithms of our IBEET-FA scheme are slightly higher than but still comparable with those of PKEET-FA scheme [8]. Figure 4 shows the computational costs of authorization algorithms of the two schemes. The running time of Type-2 and Type-3 authorization algorithms of the two schemes increase linearly with the repetition numbers. The running time of Aut_1 is constant. Although our Aut_2 algorithm has a slightly lower efficiency than that of PKEET-FA, our Aut_3 algorithm is more efficient.

Figure 5 shows the computational costs of test algorithms. Our Test_1 algorithm is slightly slower than that of

PKEET-FA, but the other test algorithms of our scheme are almost as efficient as those of PKEET-FA. Generally, our IBEET-FA scheme has almost the same computational efficiency with PKEET-FA scheme [8].

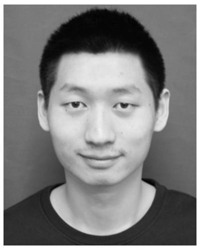
VII. CONCLUSION AND FUTURE WORK

In this paper, we introduced a new notion of identity-based encryption, called IBEET-FA, which supports flexible and authorized equality test on ciphertexts. The notion can be applied to search over ciphertexts encrypted with different public keys in IBE settings. We gave the security models of IBEET-FA and proposed a concrete construction, which was proved to be secure under the given model based on standard mathematical assumptions. Compared with its counterpart in the PKI setting, our scheme has almost the same efficiency and furthermore does not suffer from the complex key management issue.

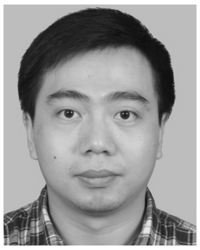
Our IBEET-FA scheme is based on bilinear pairing, which is still computationally expensive. In future work, we consider constructing IBEET-FA schemes without using bilinear pairing.

REFERENCES

[1] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Dependable Secure Comput.*, to be published.
[2] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2016, pp. 173–195. doi: 10.1007/978-3-319-45744-4_9.
[3] J. Shen, T. Zhou, X. Chen, J. Li, and W. Susilo, "Anonymous and traceable group data sharing in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 4, pp. 912–925, Apr. 2018.
[4] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2019.
[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2004, pp. 506–522.
[7] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Proc. Cryptographers's Track RSA Conf.* Berlin, Germany: Springer, 2010, pp. 119–131.
[8] 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.
[9] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*. Berlin, Germany: Springer, 1984, pp. 47–53.
[10] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable Symmetric Encryption: Designs and Challenges," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 40:1–40:37, 2017.
[11] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. 31st Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2011, pp. 273–281.
[12] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[13] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.
[14] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. Int. Workshop Inf. Secur. Appl.* Berlin, Germany: Springer, 2004, pp. 73–86.
[15] J. W. Byun, H. S. Rhee, H.-A. Park, and D.-H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. Workshop Secure Data Manage.* Berlin, Germany: Springer, 2006, pp. 75–83.
[16] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proc. Int. Conf. Autonomic Trusted Comput.* Berlin, Germany: Springer, 2008, pp. 100–105.
[17] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electron. Express*, vol. 6, no. 5, pp. 237–243, 2009.
[18] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Inf. Sci.*, vol. 265, pp. 176–188, May 2014.
[19] Q. Tang, "Towards public key encryption scheme supporting equality test with fine-grained authorization," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany: Springer, 2011, pp. 389–406.
[20] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," *Secur. Commun. Netw.*, vol. 5, no. 12, pp. 1351–1362, 2012.
[21] X.-J. Lin, H. Qu, and X. Zhang, "Public key encryption supporting equality test and flexible authorization without bilinear pairings," Cryptol. ePrint Arch., Tech. Rep. 2016/277, 2016. [Online]. Available: https://eprint.iacr.org/2016/277
[22] Y. Xu, M. Wang, H. Zhong, J. Cui, L. Liu, and V. N. L. Franqueira, "Verifiable public key encryption scheme with equality test in 5G networks," *IEEE Access*, vol. 5, pp. 12702–12713, 2017.
[23] S. Canard, G. Fuchsbauer, A. Gouget, and F. Laguillaumie, "Plaintext-checkable encryption," in *Cryptographers' Track at the RSA Conference*. Berlin, Germany: Springer, 2012, pp. 332–348.
[24] S. Ma, "Identity-based encryption with outsourced equality test in cloud computing," *Inf. Sci.*, vol. 328, pp. 389–402, Jan. 2016.
[25] 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.
[26] 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.
[27] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2001, pp. 213–229.
[28] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer, 2003, pp. 301–312.
[29] X. Boyen, "The uber-assumption family," in *Pairing-Based Cryptography—Pairing*. Berlin, Germany: Springer, Sep. 2008, pp. 39–56.
[30] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[31] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.
[32] *VMware*. (2017). [Online]. Available: http://www.vmware.com
[33] B. Lynn *et al.* (2013). *Pairing-Based Cryptography Library*. [Online]. Available: https://crypto.stanford.edu/pbc/



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.



QIONG HUANG received the B.S. and M.S. degrees from Fudan University, in 2003 and 2006, respectively, and 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. His research interests include cryptography and information security, in particular, and cryptographic protocols design and analysis.

SHA MA received the B.S. and M.S. degrees from Wuhan University, in 2004 and 2006, respectively, and the Ph.D. degree from South China Agricultural University, Guangzhou, China, in 2012, where she is currently an Associate Professor with the College of Mathematics and Informatics. Her research interests include applied cryptography and security in cloud computing.

JIAN SHEN received the M.E. and Ph.D. degrees in computer science from Chosun University, South Korea, in 2009 and 2012, respectively. Since 2012, he has been a Professor with the Nanjing University of Information Science and Technology, Nanjing, China. His current research interests include public key cryptography, secure data sharing, and data auditing in cloud.

WILLY SUSILO (SM'01) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is currently a Professor, the Head of School of Computing and Information Technology, and the Director of Institute of Cybersecurity and Cryptology with the University of Wollongong. He has authored or coauthored over 300 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. He served as a Program Committee Member in dozens of international conferences. He was a recipient of the prestigious ARC Future Fellow by the Australian Research Council and the Researcher of the Year Award in 2016 by the University of Wollongong. His work has been cited over 9000 times in Google Scholar. He is the Editor-in-Chief of the *Information* journal. He is currently serving as an Associate Editor for several international journals, including *Computer Standards and Interfaces* (Elsevier) and the *International Journal of Information Security* (Springer).