entropy

MDPI

Article

Efficient Equality Test on Identity-Based Ciphertexts Supporting Flexible Authorization

Na Li

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; lina_2014@163.com

Abstract: In the cloud, uploading encrypted data is the most effective way to ensure that the data are not leaked. However, data access control is still an open problem in cloud storage systems. To provide an authorization mechanism to limit the comparison of a user's ciphertexts with those of another, public key encryption supporting the equality test with four flexible authorizations (PKEET-FA) is presented. Subsequently, more functional identity-based encryption supporting the equality test (IBEET-FA) further combines identity-based encryption with flexible authorization. The bilinear pairing has always been intended to be replaced due to the high computational cost. Hence, in this paper, we use general trapdoor discrete log groups to construct a new and secure IBEET-FA scheme, which is more efficient. The computational cost for the encryption algorithm in our scheme was reduced to 43% of that of the scheme of Li et al. In Type 2 and 3 authorization algorithms, the computational cost of both was reduced to 40% of that of the scheme of Li et al. Furthermore, we give proof that our scheme is secure against one-wayness under the chosen identity and chosen ciphertext attacks (OW-ID-CCA), and indistinguishable against chosen identity and chosen ciphertext attacks (IND-ID-CCA).

Keywords: equality test; trapdoor discrete log groups; identity-based encryption; cloud storage

Citation: Li, N. Efficient Equality Test on Identity-Based Ciphertexts Supporting Flexible Authorization. *Entropy* **2023**, *25*, 362. https://doi.org/10.3390/e25020362

Academic Editor: Adam Lipowski

Received: 1 December 2022
Revised: 18 January 2023
Accepted: 22 January 2023
Published: 15 February 2023

1. Introduction

With the application of the Internet increasingly spreading, people have more extensive storage and computing requirements for cloud servers. Users make full use of cloud servers, allowing cloud servers to help them in storing and processing data, reducing the user's storage burden and computing overhead. Users in different regions can upload data onto and download data from a server, which provides convenience for users to share data. However, servers are also vulnerable to some attacks. If users store their data unencrypted in the cloud server, attackers or malicious internal administrators may access the data stored by users. The solution is for every user to upload encrypted data onto the cloud server. Previous classical encryption schemes cannot realize direct searches or calculations in the ciphertext.In a searchable encryption scheme [1], the ciphertext and trapdoor for retrieval need to be obtained with the same public and private key pair.

A novel PKEET scheme [2] was first proposed by Yang et al. in 2010. In this scheme, users can test whether ciphertexts encrypted by different public keys contain the same plaintext without decrypting the ciphertext, which avoids the previous limitations of searchable encryption. However, in the scheme, anyone can test the encrypted data, which can lead to data leakage. Taking into account better meeting practical applications, Tang proposed a fine-grained equality test scheme [3] that can achieve fine-grained authorization by sending tokens to a proxy. The equality test of flexible authorization for more scenarios was proposed in [4], in which there were different authorizations to meet the different needs of users, and different authorization types corresponded to different test permissions. It can not only perform the equivalence testing of ciphertext that was encrypted without the same public key, but also designate testers, which better protects the privacy of the

data. On this basis, to avoid the public key infrastructure (PKI), a functional and efficient IBEET-FA scheme [5] is proposed as a new concept, replacing PKE with IBE. The first IBE scheme [6] replaced the public key with user-related identity information, and the private key is calculated and provided by a trusted third party. No need for a public key means that the difficulty of key management is eliminated. A new IBE scheme [7] using the general trapdoor discrete logarithm group was proposed that reduces the computational cost compared to that when using bilinear pairs. IBEET-FA [5] is based on bilinear pairing.

1.1. Our Contribution

Bilinear pairing is computationally expensive, and to reduce the computational cost, we have attempted to replace pairing with discrete logarithms. We reconstructed an existing concept with a different tool, namely, reconstructing the IBEET-FA scheme with discrete logarithms. This can achieve more efficient searches in ciphertexts encrypted by different public keys, and maintain the nature of flexible authorization in which different authorizations correspond to different permissions. A public key infrastructure is not required.

We first defined the scheme and its correctness. Subsequently, a specific scheme IBEET-FA without paring was constructed, and the scheme was proven to be correct. Our scheme is communicationally efficient, and it has a small public key and ciphertext. The scheme is computationally efficient, as the Aut-1, Aut-2, and Aut-3 authorization algorithms and testing algorithms in it all have a small computational overhead.

We then define two security models for the scheme, and two types of adversaries, Adv-I and Adv-II. Our IBEET-FA without a pairing scheme achieved OW-ID-CCA security for Aut-γ (γ = 1, 2, 3) against Adv-I on the basis of the CDH assumption in the random oracle model. The IBEET-FA without a pairing scheme achieved IND-ID-CCA security for Aut-γ (γ = 1, 2, 3) against Adv-II on the basis of the DDH assumption.

1.2. Related Works

A new concept of public key encryption with keyword search (PEKS) was proposed by Boneh et al. [1] in 2004 that allows for direct keyword searches in ciphertext without decrypting the ciphertext. A user can generate the corresponding trapdoor of some keyword by using its private key and perform a keyword search in the ciphertexts with the trapdoor. Subsequently, many related variants were proposed [8–10]. Bellare et al. [11] proposed a deterministic PKE scheme. Yang et al. [2] devised a ciphertext-based equality test scheme using bilinear groups for searchable and classified encrypted data. However, in that scheme, anyone could perform the test, so it is easy for it to cause data leakage, which is not conducive to data privacy. Tang [3] presented a new method where two users could authorize a proxy to execute equality calculation on their encrypted message by issuing tokens. Tang [12] gave a new PKE in a two-proxy model supporting fine-grained authorization (FG-PKEET) in which the two proxies were required to cooperate to complete the equality test. Subsequently, Tang [13] proposed the construction of an all-or-nothing PKEET (AoN-PKEET).

A new scheme of PKE with a delegated equality test (PKE-DET) was proposed by Ma et al. in [14]; in a multiuser model, only the delegated party can perform the equality test. Wu et al. [15] introduced a new equality test concept that could achieve security against insider attacks. Ma [16] proposed a variant of PKEET in which a cloud server could directly execute the equality test on the ciphertexts of the specified user, realizing the security of the cloud database application. In [17], PKE-AET offered a new idea regarding two different kinds of warrants, namely, receiver warrants and cipher warrants. After a tester receives a receiver warrant from some receiver, the tester can perform the equality test on any of the receiver's ciphertext; in the second case, after a tester receives a cipher warrant associated with some ciphertext from some receiver, the tester can just execute an equality test on that ciphertext. Huang et al. [18] presented a ciphertext-binded authority (CBA) PKEET scheme. CBAs are only valid for specific ciphertexts, and they are invalid for

other ciphertexts encrypted by the same public key. The concept of the filtered equality test (FET) was proposed by Huang et al. [19] where the receiver selects a set of messages and generates the corresponding warrant. After a user receives the warrant, if the plaintext corresponding to the ciphertext is in the message set, they can perform an equality test on the recipient's ciphertext. Huang et al. [20] proposed a PKE-FET scheme in which FET was also applied to construct searchable encryption. The key policy-attribute-based encryption with an equality test scheme was proposed by Zhu et al. in [21]. After the flexible scheme, a ciphertext policy-attribute-based encryption scheme was presented by Wang et al. [22] that also supported the function of the equality test.

A new authorization mechanism for efficient PKEET-FA was proposed by Ma et al. [4], which can more effectively achieve user privacy protection. The scheme was based on bilinear pairing, Lin et al. [23] made improvements on this basis and proposed a novel PKEET-FA scheme, Bilinear pairings were not used in this scheme. This protocol used a quadratic curve to do the equality test, Zhu et al. [24] used a simpler straight line for the equality test. A new concept of IBEET by combining two existing concepts PKEET and IBE was given by Ma et al. [25]. A new IBEET-FA scheme was proposed in [5]. Users can directly execute equality tests on the ciphertext, eliminating the need for complex key management.

Duong et al. [26] proposed a new PKEET scheme based on ideal lattices and a scheme based on integer lattices, both schemes can achieve CCA2-security. Ref. [27] introduced the trends in multimedia forensics, and many deep-learning-based techniques. In [28], lSusilo et al. presented a novel concept of public key encryption with multi-ciphertext equality test (PKE-MET), which enables the cloud server to perform equality tests among multiple ciphertexts. A new primitive of identity-based encryption with equality test and datestamp-based authorization mechanism (IBEET-DBA) was proposed by Lin et al. [29], in which the data owner could control the valid period of trapdoor by using datestamp. Deverajan et al. [30] presented public key encryption with equality test based on discrete logarithm problem (DLP). Considering the possible attacks on trapdoors given to cloud servers and the different computing power of the entities, Vaanchig et al. [31] introduced a notion of secure-channel-free IBEET (SCF-IBEET).

1.3. Organization

We organize the remainder of the paper as follows. The definitions of Trapdoor Discrete Log Groups and Decision Diffie–Hellman Problem are given in Section 2. Then, we give the system model, the definitions of IBEET-FA and the security model in Section 3. In Section 4, we propose a new IBEET-FA scheme without pairing. In Section 5, the security analysis of our scheme will be given. In Section 6, we will show the complexity comparison of our scheme and other related schemes. In the last section, some conclusions will be given.

2. Preliminaries

2.1. Trapdoor Discrete Log (TDL) Groups

Definition 1. *A TDL group generator consists of algorithms TDLGen and SolveDL:*

- $TDLGen(k)$*: Given security parameter k as the input, the algorithm returns a tuple* (T, q, g, G) *where T is used to denote the trapdoor, q is used to denote the prime order, g is used to denote a random generator, and G is used to denote a group.*
- $SolveDL(k, (T, q, g, G), h)$*: Given the inputs of a security parameter k,* (T, q, g, G) *denoting a tuple and h denoting a group element, the algorithm outputs* $\alpha \in Z_q$*, and* $h = g^{\alpha}$ *holds.*

2.2. Computational Diffie–Hellman (CDH) Problem

Definition 2. *Let q be the prime order of group G, generator g is gotten from the running result of algorithm TDLGen in the Definition* 1*, let* (g, g^a, g^b) *be a tuple in G, for* $a, b \in Z_q$*. It is intractable to compute* g^{ab}*.* $\mathcal{A}$ *is an adversary, in probability polynomial time, the advantage of adversary* $\mathcal{A}$ *to solve the CDH problem is*

$$Adv_{\mathcal{A},G}^{CDH}(k) = P(\mathcal{A}(g, g^a, g^b) = g^{ab}, G)$$

2.3. Decision Diffie–Hellman (DDH) Problem

Definition 3. *Let q be the prime order of group G, generator g is gotten from the running result of algorithm $TDLGen$ in the $Definition$ 1, let (g, g^a, g^b, g^c), (g, g^a, g^b, g^{ab}) be two tuples in G, for $a, b, c \in Z_q$. It is difficult to distinguish the two tuples in this computational relationship. $\mathcal{A}$ is an adversary, in probability polynomial time, the advantage of $\mathcal{A}$ to solve the DDH problem is*

$$Adv_{\mathcal{A},G}^{DDH}(k) = |P(\mathcal{A}(g, g^a, g^b, g^{ab}) = 1, G) - P(\mathcal{A}(g, g^a, g^b, g^c) = 1, G)|$$

3. System Model and Definition

In Sections 3.1 and 3.2, we give the system model and the definition of IBEET-FA, similarly in [5]. In Section 3.3, we give the security model of IBEET-FA.

3.1. System Model

In our defined IBEET-FA scheme, we give four entities: a cloud server, a trusted third party, and two users labeled as i and j. The trusted third party generates system parameters for users and cloud service. User i and user j encrypt their data with their public key, and store ciphertext in the cloud server, and the cloud server is authorized to do equality tests on stored ciphertext, but the server does not have the ability to decrypt them. We present the IBEET-FA system model in Figure 1.

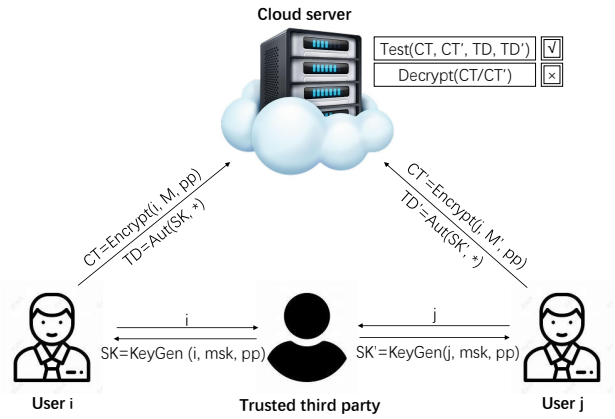

Figure 1. IBEET-FA system model.

3.2. Definition of IBEET-FA

Definition 4. *Our IBEET-FA scheme consists of four algorithms:*

- *$Setup(k)$: Taken security parameter k as the input, the public parameter pp and the master secret key msk will be gotten from the running result of the algorithm.*
- *$KeyGen(i, msk, pp)$: Given label i, master secret key msk, and public parameter pp as input, the algorithm returns the secret key $SK = (\alpha_i, \beta_i)$.*
- *$Encrypt(i, M, pp)$: Given the inputs of user i, a message M and public parameter pp, the algorithm returns the ciphertext CT.*
- *$Decrypt(i, \alpha_i, CT, pp)$: Given label i, a private key α_i, a ciphertext CT and public parameter pp as inputs, a message M will be gotten from the running result of the algorithm, or returns an error symbol $\perp$.*

User i has the public-secret key pair (i, SK), corresponding encrypted data is CT, User j has the public-secret key pair $(j, SK^{'})$, corresponding encrypted data is $CT^{'}$. They have four types of authorization, corresponding to four different Aut algorithms and four different $Test$ algorithms. Aut algorithm is used to generate trapdoors for users, and the cloud service runs $Test$ procedure to test whether or not two different encrypted data contain the same message.

Aut-1:

- $Aut_1(i, SK)$: Given user i and i's secret key SK as inputs, the authorization procedure returns a trapdoor TD_1.
- $Test_1(CT, CT^{'}, TD_1, TD_1^{'})$: Given the inputs of i'ciphertext CT, i'trapdoor TD_1, j'ciphertext $CT^{'}$ and j'trapdoor $TD_1^{'}$, the test procedure returns 1 if two ciphertexts contain the same message, otherwise returns 0.

Aut-2:

- $Aut_2(SK, CT)$: Given the inputs of user i'private key SK and a ciphertext CT, the authorization procedure outputs a trapdoor TD_2.
- $Test_2(CT, CT^{'}, TD_2, TD_2^{'})$: Given the inputs of i'ciphertext CT, i'trapdoor TD_2, j'ciphertext $CT^{'}$ and j'trapdoor $TD_2^{'}$, the test procedure returns 1 if two ciphertexts contain the same plaintext, otherwise returns 0.

Aut-3:

- $Aut_3(SK, CT, CT^{'})$: Given the inputs of user i'private key SK, i'ciphertext CT, and j'ciphertext $CT^{'}$, the authorization procedure outputs a trapdoor TD_3.
- $Test_3(CT, CT^{'}, TD_3, TD_3^{'})$: Given the inputs of i'ciphertext CT, i'trapdoor TD_3, j'ciphertext $CT^{'}$ and j'trapdoor $TD_3^{'}$, the test procedure returns 1 if two ciphertexts contain the same plaintext, otherwise returns 0.

Aut-4:

- $Aut_4(SK, CT)$: Given the inputs of user i'private key SK and ciphertext CT, the authorization procedure returns a trapdoor TD_4.
- $Aut_4(j, SK^{'})$: Given user j and j's secret key $SK^{'}$ as inputs, the authorization procedure returns a trapdoor $TD_4^{'}$.
- $Test_4(CT, CT^{'}, TD_4, TD_4^{'})$: Given the inputs of i'ciphertext CT, i'trapdoor TD_4, j'ciphertext $CT^{'}$ and j'trapdoor $TD_4^{'}$, the test procedure returns 1 if two ciphertexts contain the same message, otherwise returns 0.

Definition 5. *(Correctness): If for any* $(msk, pp) \leftarrow Setup(k)$, $(\alpha_i, \beta_i) \leftarrow KeyGen(msk, pp, i)$, $(\alpha_j, \beta_j) \leftarrow KeyGen(msk, pp, j)$, *the following conditions can be satisfied, we say an IBEET-FA scheme is correct.*

1. For any possible plaintext M in the plaintext space, $Decrypt(Encrypt(M, i, pp), pp, \alpha_i, i) = M$, all equations hold.
2. For any possible ciphertext CT of user i and any possible ciphertext $CT^{'}$ of user j, if Decrypt$(i, \alpha_i, CT, pp) = Decrypt(j, \alpha_j, CT^{'}, pp) \neq \perp$:
   Aut-1: For two trapdoors of $Aut_1(i, SK) = TD_1$, $Aut_1(j, SK^{'}) = TD_1^{'}$, the following equality always holds that

$$Test_1(CT, TD_1, CT^{'}, TD_1^{'}) = 1.$$

   Aut-2: For two trapdoors of $Aut_2(SK, CT) = TD_2$, $Aut_2(SK^{'}, CT^{'}) = TD_2^{'}$, the following equality always holds that

$$Test_2(CT, TD_2, CT^{'}, TD_2^{'}) = 1.$$

   Aut-3: For two trapdoors of $Aut_3(SK, CT, CT^{'}) = TD_3$, $Aut_3(SK^{'}, CT^{'}, CT) = TD_3^{'}$, the following equality always holds that

$$Test_3(CT, TD_3, CT^{'}, TD_3^{'}) = 1.$$

Aut-4: For two trapdoors of $Aut_4(SK, CT) = TD_4$, $Aut_4(j, SK^{'}) = TD_4^{'}$, the following equality always holds that

$$Test_4(CT, TD_4, CT^{'}, TD_4^{'}) = 1.$$

3. For any possible ciphertext CT of user i and any possible ciphertext $CT^{'}$ of user j, if Decrypt$(i, \alpha_i, CT, mpk) \neq Decrypt(j, \alpha_j, CT^{'}, mpk)$, where $\epsilon(\cdot)$ be a negligible function about k:
Aut-1: For two trapdoors of $Aut_1(i, SK) = TD_1$, $Aut_1(j, SK^{'}) = TD_1^{'}$, the following equality always holds that

$$P[Test_1(CT, TD_1, CT^{'}, TD_1^{'}) = 1] \leq \epsilon(k).$$

Aut-2: For two trapdoors of $Aut_2(SK, CT) = TD_2$, $Aut_2(SK^{'}, CT^{'}) = TD_2^{'}$, the following equality always holds that

$$P[Test_2(CT, TD_2, CT^{'}, TD_2^{'}) = 1] \leq \epsilon(k).$$

Aut-3: For two trapdoors of $Aut_3(SK, CT, CT^{'}) = TD_3$, $Aut_3(SK^{'}, CT^{'}, CT) = TD_3^{'}$, the following equality always holds that

$$P[Test_3(CT, TD_3, CT^{'}, TD_3^{'}) = 1] \leq \epsilon(k).$$

Aut-4: For two trapdoors of $Aut_4(SK, CT) = TD_4$, $Aut_4(j, SK^{'}) = TD_4^{'}$, the following equality always holds that

$$P[Test_4(CT, TD_4, CT^{'}, TD_4^{'}) = 1] \leq \epsilon(k).$$

3.3. Security Model

According to the nature of our scheme, we use the IBEET-FA security models defined in [5]. Since Aut-4 is a combination of one user authorization way in Aut-1 and one user authorization way in Aut-2, we omit Aut-4 authorization queries for simplicity. Adversaries are only allowed to query for Aut-γ (γ = 1, 2, 3). We define two kinds of adversaries for the security model of our IBEET-FA scheme:

- Adv-I: For Aut-γ (γ = 1, 2, 3), with Aut-γ trapdoor information, the adversary can not get the plaintext from the challenge ciphertext.
- Adv-II: For Aut-γ (γ = 1, 2, 3), without Aut-γ trapdoor information, the adversary can not know the challenge ciphertext is from which plaintext.

Under chosen ciphertext and chosen identity attacks, We now define the one-wayness security (OW-ID-CCA) against Adv-I for Aut-γ (γ = 1, 2, 3) as follows:

GameI: Let the receiver have index t ($1 \leq t \leq n$), and assume $\mathcal{A}_1$ is a Adv-I. Between the challenger $\mathcal{C}_1$ and the adversary $\mathcal{A}_1$, the game goes as follows:

- *Setup*: Challenger $\mathcal{C}_1$ firstly picks k as a security parameter, then gets public parameter pp by calling $Setup$ algorithm, sends pp to $\mathcal{A}_1$.
- *Phase*1: Allows $\mathcal{A}_1$ to query for polynomially many times as follows.
  1. Key retrieve queries: $\mathcal{C}_1$ calls $KeyGen(i, pp, msk)$ algorithm and sends SK to $\mathcal{A}_1$. call the algorithm and send the result to A
  2. Decryption queries: $\mathcal{C}_1$ runs $Decrypt(pp, CT, \alpha_i, i)$ algorithm and returns M(which might be $\perp$) to $\mathcal{A}_1$.
  3. Authorization queries: For three types of authorization Aut-γ (γ = 1, 2, 3),
     (a) i as input, $\mathcal{C}_1$ sends TD_1 to $\mathcal{A}_1$.
     (b) (i, CT) as input, $\mathcal{C}_1$ sends TD_2 to $\mathcal{A}_1$.
     (c) $(i, CT, j, CT^{'})$ as input, $\mathcal{C}_1$ sends TD_3 to $\mathcal{A}_1$.

- *Challenge*: Adversary $\mathcal{A}_1$ picks a target identity t which has not been queried in extract queries, and sends it to $\mathcal{C}_1$. Then $\mathcal{C}_1$ chooses a message M_t randomly, gets $C_t^* = Encrypt(M_t, t, pp)$ as the challenge ciphertext and sends it to $\mathcal{A}_1$.
- *Phase*2: $\mathcal{A}_1$ continues issuing the same query as Phase 1. However, t can not be queried in this phase and (t, C_t^*) can not be queried in a decryption query.
- *Guess*: $\mathcal{A}_1$ returns a message M', if $M' = M_t$ means $\mathcal{A}_1$ wins the game.

We give the advantage definition of $\mathcal{A}_1$ in the Game I as

$$Adv_{IBEET-FA,\mathcal{A}_1}^{OW-ID-CCA,Aut-\gamma}(k) = P[M' = M_t](\gamma = 1, 2, 3).$$

Definition 6. *If the advantage* $Adv_{IBEET-FA,\mathcal{A}_1}^{OW-ID-CCA,Aut-\gamma}(k)$ *is negligible for any probabilistic polynomial-time Adv-I* $\mathcal{A}_1$*, We say the IBEET-FA scheme is OW-ID-CCA secure for three types of authorization Aut-*γ *(*γ *= 1, 2, 3).*

GameII: Let the recipient's identity be t ($1 \leq t \leq n$), and Sets $\mathcal{A}_2$ as an Adv-II adversary. Between the challenger $\mathcal{C}_2$ and the adversary $\mathcal{A}_2$ the game goes as follows:

- *Setup*: Challenger $\mathcal{C}_2$ firstly picks k as a security parameter, then gets public parameter pp by calling $Setup$ algorithm, and sends pp to $\mathcal{A}_2$.
- *Phase*1: Allows $\mathcal{A}_2$ to issue polynomially times queries as in Game I.
- *Challenge*: Adversary $\mathcal{A}_2$ sends to Challenger $\mathcal{C}_2$ two messages M_0, M_1, and a target identity t, t can not be allowed to appear in extract query or Aut-1 authorization query phase. $\mathcal{C}_2$ picks a bit $b \in \{0, 1\}$ randomly, uses encryption algorithm to get challenge ciphertext $C^* = Encrypt(M_b, t, pp)$, then sends C^* to $\mathcal{A}_2$.
- *Phase*2: Allows $\mathcal{A}_2$ to continue issuing queries as Phase 1, but there are some restrictions as follows:
  1. i can not be queried in the Key retrieve query or Aut-1 authorizations queries;
  2. (i, C^*) can not be queried in the decryption query;
  3. (i, C^*) can not be queried in the authorizations query.
- *Guess*: $\mathcal{A}_2$ returns a bit b', when $b' = b$ holds, $\mathcal{A}_2$ wins in the game.

In Game II, the advantage definition of $\mathcal{A}_2$ is

$$Adv_{IBEET-FA,\mathcal{A}_2}^{IND-ID-CCA,Aut-\gamma}(k) = |P[b' - b] - \frac{1}{2}|(\gamma = 1, 2, 3).$$

Definition 7. *If the advantage* $Adv_{IBEET-FA,\mathcal{A}_2}^{IND-ID-CCA,Aut-\gamma}(k)$ *is negligible for any probabilistic polynomial-time Adv-II* $\mathcal{A}_2$*, We say the IBEET-FA scheme is IND-ID-CCA secure for three types of authorization Aut-*γ *(*γ *= 1, 2, 3).*

4. Our Proposed IBEET-FA Scheme

In our IBEET-FA scheme, we use the advantages of the PKEET-FA scheme and IBE without pairing scheme.

4.1. The Proposed Scheme

- $Setup(k)$: Here k is a security parameter, and it is the size of plaintext messages, the algorithm works as follows:
  1. This algorithm calls the TDLGen algorithm of the TDL generator, then gets a tuple (T, G, g, q) where T is the trapdoor, G is a group, g is a random generator, and q is the prime order.
  2. Picks some secure hash functions: $H, H_1 : \{0,1\}^* \rightarrow G$, $H_2 : G \rightarrow \{0,1\}^k$, $H_3, H_4 : \{0,1\}^k \rightarrow Z_q$, and $H_5 : G^3 \rightarrow {Z_q}^2$.
     Gets the master secret key $msk = T$, the public parameter $pp = \{H, H_1, H_2, H_3, H_4, H_5, G, g, q, k\}$.

- $KeyGen(i, pp, msk)$: Choosing label i, the public parameter pp and master secret key msk as input, then calls SolveDL algorithm. $H(i)$ as input, get a value $\alpha_i \in Z_q$ such that $g^{\alpha_i} = H(i)$. Furthermore, calls SolveDL algorithm again taking $H_1(i)$ as input to get a value $\beta_i \in Z_q$ such that $g^{\beta_i} = H_1(i)$. Then outputs the secret key $sk_i = (\alpha_i, \beta_i)$.
- $Encrypt(M, i, pp)$: Taking a plaintext M, public parameter pp and user i as input, the algorithm works as follows:
  1. Compute one point $P = (H_3(M), H_4(M))$.
  2. O is the origin, use point P, O to construct a ray $f(x)$ with O as the endpoint.
  3. Choose a non zero point $x_i \in \{0,1\}^l$, then compute $y_i = f(x_i)$.
  4. Choose at random $r \in {Z_q}^*$, then compute

$$\begin{aligned} CT_1 &= g^r, \\ CT_2 &= M \oplus H_2(H(i)^r), \\ CT_3 &= (x_i \| y_i) \oplus H_5(H_1(i)^r, CT_1, CT_2). \end{aligned}$$

  Output the ciphertext $CT = (CT_1, CT_2, CT_3)$.
- $Decrypt(i, CT, SK, pp)$: Taking label i, a ciphertext CT, private key SK and public parameter pp as input, this algorithm computes $M = CT_2 \oplus H_2(CT_1^{\alpha_i})$ and $x_i \| y_i = CT_3 \oplus H_5(CT_1^{\beta_i}, CT_1, CT_2)$. Obtain point P as in *Encrypt* algorithm and obtain $f(x)$ with P, O as in *Encrypt* algorithm. if $y_i = f(x_i)$ hold, then returns M; and returns an error symbol $\perp$ otherwise.

Two users are represented as u_i and u_j, selecting r_i and r_j as the randomness used in computing CT and $CT^{'}$. Correspondingly, compute ciphertext $CT = (CT_1, CT_2, CT_3)$ and ciphertext $CT^{'} = (CT_1^{'}, CT_2^{'}, CT_3^{'})$ of u_i and u_j.

Aut-1:

- $Aut_1(i, SK)$: This authorization procedure returns a trapdoor $TD_1 = \beta_i$.
- $Test_1(CT, CT^{'}, TD_1, TD_1^{'})$: The test procedure performs the following calculations

$$\begin{aligned} x_i \| y_i &= CT_3 \oplus H_5({CT_1}^{TD_1}, CT_1, CT_2), \\ x_j \| y_j &= CT_3^{'} \oplus H_5({CT_1^{'}}^{TD_1^{'}}, CT_1^{'}, CT_2^{'}). \end{aligned}$$

  It returns 1 if $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, or returns 0 otherwise.

Aut-2:

- $Aut_2(SK, CT)$: This authorization procedure outputs a trapdoor $TD_2 = H_5({CT_1}^{\beta_i}, CT_1, CT_2)$.
- $Test_2(CT, CT^{'}, TD_2, TD_2^{'})$: This test procedure computes

$$\begin{aligned} x_i \| y_i &= CT_3 \oplus TD_2, \\ x_j \| y_j &= CT_3^{'} \oplus TD_2^{'}. \end{aligned}$$

  It returns 1 if $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, or returns 0 otherwise.

Aut-3:

- $Aut_3(SK, CT, CT^{'})$: This authorization procedure recovers y_i with SK, then outputs a trapdoor

$$\begin{aligned} TD_3 &= (TD_{i,1}, TD_{i,2}) \\ &= ([H_2({CT_1}^{\beta_i}, CT_1, CT_2)]_0^{l-1}, (CT_1 CT_1^{'})^{y_i}). \end{aligned}$$

- $Test_3(CT, CT^{'}, TD_3, TD_3^{'})$: This test procedure computes

$$x_i = [CT_1]_0^{l-1} \oplus TD_{i,1},$$
$$x_j = [CT_1^{'}]_0^{l-1} \oplus TD_{j,1}.$$

It returns 1 if ${TD_{i,2}}^{\frac{1}{x_i}} = {TD_{j,2}}^{\frac{1}{x_j}}$, or returns 0 otherwise.

Aut-4 :

- $Aut_4(SK, CT)$: This authorization procedure returns a trapdoor $TD_4 = TD_2 = H_5({CT_1}^{\beta_i}, CT_1, CT_2)$.
- $Aut_4(j, SK^{'})$: This authorization procedure returns a trapdoor $TD_4^{'} = TD_1^{'} = \beta_j$.
- $Test_4(CT, CT^{'}, TD_4, TD_4^{'})$: This test procedure computes

$$x_i \| y_i = CT_3 \oplus TD_4,$$
$$x_j \| y_j = CT_3^{'} \oplus H_5({CT_1^{'}}^{TD_4^{'}}, CT_1^{'}, CT_2^{'}).$$

It returns 1 if $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, or returns 0.

4.2. Correctness

Theorem 1. *By definition 2, the correctness of the above IBEET-FA scheme is proven.*

Proof of Theorem 1. We now prove our IBEET-FA scheme meets all correctness requirements.

1. The first requirement is satisfied obviously.
2. According to the second requirement, for any $(\alpha_i, \beta_i) \leftarrow KeyGen(msk, pp, i)$, $(\alpha_j, \beta_j) \leftarrow KeyGen(msk, pp, j)$, $CT = (CT_1, CT_2, CT_3)$ $= Encrypt(M_i, i, pp)$, $CT^{'} = (CT_1^{'}, CT_2^{'}, CT_3^{'}) = Encrypt(M_j, j, pp)$, all the following equations hold.
   - Aut-1: Given $TD_1 = \beta_i$, $TD_1^{'} = \beta_j$, get the following:

$$x_i \| y_i = CT_3 \oplus H_5({CT_1}^{TD_1}, CT_1, CT_2),$$
$$x_j \| y_j = CT_3^{'} \oplus H_5({CT_1^{'}}^{TD_1^{'}}, CT_1^{'}, CT_2^{'}).$$

     Because point (x_i, y_i) is taken from the ray corresponding to M_i, point (x_j, y_j) is taken from the ray corresponding to M_j, if $M_i = M_j$ means (x_i, y_i) and (x_j, y_j) are taken from the same ray. So $\frac{y_i}{x_i} = \frac{y_j}{x_j}$ holds if $M_i = M_j$.
   - Aut-2: Given

$$TD_2 = H_5({CT_1}^{\beta_i}, CT_1, CT_2)$$

     and

$$TD_2^{'} = H_5({CT_1^{'}}^{\beta_j}, CT_1^{'}, CT_2^{'})$$

     get the following:

$$x_i \| y_i = CT_3 \oplus TD_2,$$
$$x_j \| y_j = CT_3^{'} \oplus TD_2^{'}.$$

     Because point (x_i, y_i) is taken from the ray corresponding to M_i, point (x_j, y_j) is taken from the ray corresponding to M_j, if $M_i = M_j$ means (x_i, y_i) and (x_j, y_j) are taken from the same ray. So $\frac{y_i}{x_i} = \frac{y_j}{x_j}$ holds if $M_i = M_j$.

- Aut-3: Given

$$TD_3 = (TD_{i,1}, TD_{i,2}) = ([H_2(CT_1{}^{\beta_i}, CT_1, CT_2)]_0^{l-1}, (CT_1 CT_1^{'})^{y_i})$$

and

$$TD_3^{'} = (TD_{j,1}, TD_{j,2}) = ([H_2(CT_1^{'}{}^{\beta_j}, CT_1^{'}, CT_2^{'})]_0^{l-1}, (CT_1^{'} CT_1)^{y_j}),$$

get the following:

$$\begin{aligned} x_i &= [C_{i,3}]_0^{l-1} \oplus TD_{i,1}, \\ x_j &= [C_{j,3}]_0^{l-1} \oplus TD_{j,1}, \\ TD_{i,2}{}^{\frac{1}{x_i}} &= ((C_{i,1}C_{j,1})^{y_i})^{\frac{1}{x_i}} = (C_{i,1}C_{j,1})^{\frac{y_i}{x_i}}, \\ TD_{j,2}{}^{\frac{1}{x_j}} &= ((C_{j,1}C_{i,1})^{y_j})^{\frac{1}{x_j}} = (C_{j,1}C_{i,1})^{\frac{y_j}{x_j}}. \end{aligned}$$

Because point (x_i, y_i) is taken from the ray corresponding to M_i, point (x_j, y_j) is taken from the ray corresponding to M_j, if $M_i = M_j$ means (x_i, y_i) and (x_j, y_j) are taken from the same ray. So $TD_{i,2}{}^{\frac{1}{x_i}} = TD_{j,2}{}^{\frac{1}{x_j}}$, i.e., $\frac{y_i}{x_i} = \frac{y_j}{x_j}$ holds if $M_i = M_j$.

- Aut-4 : Given $TD_4 = TD_2 = H_5(CT_1{}^{\beta_i}, CT_1, CT_2)$, $TD_4^{'} = TD_1^{'} = \beta_j$, get the following:

$$\begin{aligned} x_i \| y_i &= CT_3 \oplus TD_4, \\ x_j \| y_j &= CT_3^{'} \oplus H_5(CT_1^{'}{}^{TD_4^{'}}, CT_1^{'}, CT_2^{'}). \end{aligned}$$

Because point (x_i, y_i) is taken from the ray corresponding to M_i, point (x_j, y_j) is taken from the ray corresponding to M_j, if $M_i = M_j$ means (x_i, y_i) and (x_j, y_j) are taken from the same ray. So $\frac{y_i}{x_i} = \frac{y_j}{x_j}$ holds if $M_i = M_j$.

3. Now we prove the third condition holds.
$F_i(x)$ is a ray passing through point $P_i = (H_3(M_i), H_4(M_i))$ with O as its endpoint, $f_j(x)$ is a ray passing through $P_j = (H_3(M_j), H_4(M_j))$ with O as its endpoint. Point (x_i, y_i) is taken from the ray $f_i(x)$, and point (x_j, y_j) is taken from the ray $f_j(x)$.

- Aut-1: If $Test_1(CT, CT^{'}, TD_1, TD_1^{'}) = 1$, we can get that $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, that is, point (x_i, y_i) and point (x_j, y_j) are taken from the same ray with O as the end point. For $M_i \neq M_j$, $P[\frac{H_4(M_i)}{H_3(M_i)} = \frac{H_4(M_j)}{H_3(M_j)}]$ is negligible, then we get that $P[Test_1(CT, CT^{'}, TD_1, TD_1^{'}) = 1]$ is also negligible for $M_i \neq M_j$.
- Aut-2: If $Test_2(CT, CT^{'}, TD_2, TD_2^{'}) = 1$, we can get that $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, that is, point (x_i, y_i) and point (x_j, y_j) are taken from the same ray with O as the end point. For $M_i \neq M_j$, $P[\frac{H_4(M_i)}{H_3(M_i)} = \frac{H_4(M_j)}{H_3(M_j)}]$ is negligible , then we get that $P[Test_2(CT, CT^{'}, TD_2, TD_2^{'}) = 1]$ is also negligible.
- Aut-3: If $Test_3(CT, CT^{'}, TD_3, TD_3^{'}) = 1$, we can get that $TD_{i,2}{}^{\frac{1}{x_i}} = TD_{j,2}{}^{\frac{1}{x_j}}$, that is, $(C_{i,1}C_{j,1})^{\frac{y_i}{x_i}} = (C_{j,1}C_{i,1})^{\frac{y_j}{x_j}}$. For $M_i \neq M_j$, $P[\frac{H_4(M_i)}{H_3(M_i)} = \frac{H_4(M_j)}{H_3(M_j)}]$ is negligible , we get that $P[Test_3(CT, CT^{'}, TD_3, TD_3^{'}) = 1]$ is also negligible for $M_i \neq M_j$.
- Aut-4: If $Test_4(CT, CT^{'}, TD_4, TD_4^{'}) = 1$, we can get that $\frac{y_i}{x_i} = \frac{y_j}{x_j}$, that is, point (x_i, y_i) and point (x_j, y_j) are taken from the same ray with O as the end point. For

$M_i \neq M_j$, $P[\frac{H_4(M_i)}{H_3(M_i)} = \frac{H_4(M_j)}{H_3(M_j)}]$ is negligible , we get that $P[Test_4(CT, CT^{'}, TD_4, TD_4^{'}) = 1]$ is also negligible for $M_i \neq M_j$.

□

5. Security Analysis

We will prove two kinds of security against different adversaries in this section. For this purpose, we design several related games to connect the scheme security and the hardness problems. Suppose $\mathcal{A}$ is a polynomial-time adversary, allowing $\mathcal{A}$ to do at most q_H, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} times of queries to hash oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{H_5}$,respectively, q_K times key generation queries, q_D times decryption queries, q_T times trapdoor queries. Challenger $\mathcal{C}$ controls oracles and answers the queries of adversaries. L_H, L_{H_1}, L_{H_2}, L_{H_3}, L_{H_4}, L_{H_5} stand for hash lists.

5.1. OW-ID-CCA Security Against Adv-I

Theorem 2. *Based on CDH assumption, in the random oracle model our presented IBEET-FA scheme is OW-ID-CCA secure against Adv-I for Aut-γ (γ = 1, 2, 3) authorization.*

Proof of Theorem 2. We design several related games to prove OW-ID-CCA security against Adv-I $\mathcal{A}_1$. Let P[$Game_i$] present the probability of breaking game i, where $i \in \{1, 2, 3\}$.
*Game*1:

- *Setup*: The challenger $\mathcal{C}_1$ outputs public parameter $\{G, g, q, k\}$, the master secret key $msk = T$.
- *Phase*1: Allows$\mathcal{A}_1$ to do the following queries.
  1. Hash queries: Suppose $\mathcal{A}_1$ queries at most q_H, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} times to hash oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$, $\mathcal{O}_{H_2}$, $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$, $\mathcal{O}_{H_5}$, respectively.
     (a) $\mathcal{O}_H$, $\mathcal{O}_{H_1}$: Set original empty lists L_H(resp.L_{H_1}). For an identity i, the oracle picks $r_{i_1} \leftarrow \mathbb{Z}_q$(resp.$r_{i_2} \leftarrow \mathbb{Z}_q$) randomly, computes$H(i) = g^{r_{i_1}}$(resp.$H_1(i) = g^{r_{i_2}}$) and records the tuple $(i, r_{i_1}, g^{r_{i_1}})$(resp.$(i, r_{i_2}, g^{r_{i_2}})$) on hash list L_H(resp. L_{H_1}). $H(i)$(resp.$H_1(i)$) is returned to $\mathcal{A}_1$.
     (b) $\mathcal{O}_{H_2}$: Set original empty lists L_{H_2}. For an input U_i, the oracle picks a string $S_i \in \{0,1\}^k$ randomly and records the tuple (U_i, S_i) on hash list L_{H_2}. $H_2(U_i) = S_i$ is returned to $\mathcal{A}_1$.
     (c) $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$: Set original empty lists L_{H_3}. For an input S_i, the oracle picks $r_i \leftarrow \mathbb{Z}_q$ randomly and records the tuple (S_i, r_i) on hash list L_{H_3}. $H_3(S_i) = r_i$ is returned to $\mathcal{A}_1$.
     (d) $\mathcal{O}_{H_5}$: Set original empty lists L_{H_5}. For an input U_i, the oracle picks a string $S_i \in \{0,1\}^{2l}$ randomly and records the tuple (U_i, S_i) on hash list L_{H_5}. $H_5(U_i) = S_i$ is returned to $\mathcal{A}_1$.
  2. Key retrieve queries: For an identity i, challenger $\mathcal{C}_1$ invokes hash oracles $\mathcal{O}_H$, $\mathcal{O}_{H_1}$ to get hash values $H(i)$, $H_1(i)$, then runs $KeyGen(msk, pp, i)$ algorithm to get the secret key $sk_i = (\alpha_i, \beta_i)$. It returns sk_i to $\mathcal{A}_1$.
  3. Decryption queries: For an identity i, ciphertext C_i, challenger $\mathcal{C}_1$ invokes key retrieve queries to obtain the secret key $sk_i = (\alpha_i, \beta_i)$, then uses sk_i to call $Decrypt(pp, C_i, \alpha_i, i)$ algorithm to obtain the message M_i(which might be $\perp$). It returns M_i (or $\perp$) to $\mathcal{A}_1$.
  4. Authorization queries: For Aut-γ (γ = 1, 2, 3),
     (a) $\gamma = 1$: i as the input, $\mathcal{C}_1$ runs Aut_1 algorithm with SK, then returns $TD_1 = \beta_i$ to $\mathcal{A}_1$.
     (b) $\gamma = 2$: (i, CT) as the input, $\mathcal{C}_1$ runs Aut_2 algorithm with SK, then returns $TD_2 = H_5({CT_1}^{\beta_i}, CT_1, CT_2)$ to $\mathcal{A}_1$.

(c) $\gamma = 3$: $(i, CT, j, CT^{'})$ as the input , $\mathcal{C}_1$ runs Aut_3 algorithm with SK, then returns $TD_3 = ([H_2(CT_1{}^{\beta_i}, CT_1, CT_2)]_0^{l-1}, (CT_1 CT_1^{'})^{y_i})$ to $\mathcal{A}_1$.

- *Challenge*: Adversary $\mathcal{A}_1$ submits to $\mathcal{C}_1$ an identity t , and t has not been queried in previous extract query, $\mathcal{C}_1$ randomly selects a message M_t, and gets $C_t^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*)$ with the following equations.

$$\begin{aligned} C_{t,1}^* &= g^{r_t}, \\ C_{t,2}^* &= M_t \oplus H_2(H(t)^{r_t}), \\ C_{t,3}^* &= (x_t \| y_t) \oplus H_5(H_1(t)^{r_t}, C_{t,1}^*, C_{t,2}^*), \end{aligned}$$

  where the point (x_t, y_t) is randomly taken from the ray passing through the point $(H_3(M_t), H_4(M_t))$, and $r_t \in {Z_q}^*$. Then, the challenge ciphertext C_t^* is sent to $\mathcal{A}_1$.
- *Phase*2: Allows $\mathcal{A}_1$ to issue the same type query as in Phase 1. However, in the key retrieve queries, t can not be allowed to query; and in the decryption queries, (t, C_t^*) can not be queried.
- *Guess*: $\mathcal{A}_1$ returns a message M', if $M' = M_t$, means in the game $\mathcal{A}_1$ wins. The probability of adversary $\mathcal{A}_1$ winning the game is:

$$Adv_{IBEET-FA,\mathcal{A}_1}^{OW-ID-CCA,Aut-\gamma}(k) = P[Game_1](\gamma = 1, 2, 3).$$

*Game*2: It is almost equivalent to Game 1, the modified parts are as follows:

$$\begin{aligned} C_{i,1} &= g^r, \\ C_{i,2} &= M \oplus R, \\ C_{i,3} &= (x_i \| y_i) \oplus H_5(H_1(i)^r, C_{i,1}^*, C_{i,2}^*). \end{aligned}$$

The change is that $H_2(H(t)^{r_t})$ is replaced by a random R. We can see that $H_2(H(t)^{r_t})$ is random in *Game*1. If $H(t)^{r_t}$ has been queried in *Game*2, we call it event E. If $H(t)^{r_t}$ has not been queried, it is difficult for $\mathcal{A}_1$ to separate *Game*1 and *Game*2. We get that

$$|P[Game1] - P[Game2]| \leq P[E],$$

then have

$$P[Game1] \leq P[Game2] + P[E].$$

Obviously, $P[E]$ is ignorable if the CDH problem is difficult.
*Game*3: It is almost equivalent to *Game*2, the modified parts are as follows:

$$\begin{aligned} C_{i,1} &= g^r, \\ C_{i,2} &= R_1, \\ C_{i,3} &= (x_i \| y_i) \oplus H_5(H_1(i)^r, C_{i,1}^*, C_{i,2}^*). \end{aligned}$$

Compared to *Game*2, $M \oplus R$ in *Game*3 is changed by random R_1. R is a random string, we can konw that $M \oplus R$ is also a random string. So it is difficult for $\mathcal{A}_1$ to separate *Game*2 and *Game*3. We have that

$$P[Game2] = P[Game3]$$

Similarly, if CDH problem is difficult, $P[Game3]$ is ignorable.

From all the formulas obtained above, we derive the following formula

$$P[Game1] \leq P[Game2] + P[E] \leq P[Game3] + P[E]$$

We can get a conclusion: when the CDH problem is intractable, our new IBEET-FA scheme can achieve IND-ID-CCA security against Adv-I. □

5.2. IND-ID-CCA Security Against Adv-II

Theorem 3. *Based on DDH assumption, in the random oracle model our presented IBEET-FA scheme is IND-ID-CCA secure against Adv-II for Aut-γ (γ = 1, 2, 3) authorization.*

Proof of Theorem 3. If such an adversary $\mathcal{A}_2$ exists who could attack the IND-ID-CCA security of this scheme, we then can get an algorithm to solve the DDH problem in polynomial time with not negligible advantage. For Adv-II $\mathcal{A}_2$, we design the following game to prove the IND-ID-CCA security. The probability of winning the game is expressed as $P[Game]$.

For $a, b, c \in Z_q$, given two tuples $(g, g^a, g^b, g^{ab}), (g, g^a, g^b, g^c) \in G$, $\mathcal{C}_2$ computes system parameters and sends to $\mathcal{A}_2$.For the queries of $\mathcal{A}_2$, $\mathcal{C}_2$ replies as following.

- *Setup*: For $i \in [1, n]$, algorithm $\mathcal{C}_2$ generates n key pairs (sk_i, pk_i), where sets $(sk_i, pk_i) = ((\alpha_i, \beta_i), (g^{\alpha_i}, g^{\beta_i}))(\alpha_i, \beta_i \in Z_q)$.
- *Phase*1: Allows algorithm $\mathcal{C}_2$ to issue four types of queries as follows.
  1. Hash queries:
     (a) $\mathcal{O}_H$, $\mathcal{O}_{H_1}$: Work in the same way as in *Game*1.
     (b) $\mathcal{O}_{H_2}$: Works in the same way as in *Game*1.
     (c) $\mathcal{O}_{H_3}$, $\mathcal{O}_{H_4}$: Works in the same way as in *Game*1.
     (d) $\mathcal{O}_{H_5}$: Works in the same way as in *Game*1.
  2. Key retrieve queries: Given an identity i, $\mathcal{C}_2$ searches tuple$(i, r_{i_1}, g^{r_{i_1}})$ and tuple $(i, r_{i_2}, g^{r_{i_2}})$ in list L_H and list L_{H_1}, sends (r_{i_1}, r_{i_2}) to $\mathcal{A}_2$ when $i \neq t$ holds. Otherwise, $\mathcal{C}_2$ returns $\perp$ to $\mathcal{A}_2$.
  3. Decryption queries: For identity i and a query ciphertext C_i, challenger $\mathcal{C}_2$ searches tuple(U, S) in list L_{H_2}, and computes $M \parallel R = C_2 + S$. If exists R, making equation $C_1 = g^R$ true, $\mathcal{C}_2$ returns M to $\mathcal{A}_2$. Otherwise, $\mathcal{C}_2$ returns $\perp$ to $\mathcal{A}_2$.
  4. Authorization queries: For Aut-γ (γ = 1, 2, 3),
     (a) $\gamma = 1$: i as the input, challenger $\mathcal{C}_2$ calls Aut_1 algorithm with SK, then sends $TD_1 = \beta_i$ to $\mathcal{A}_2$.
     (b) $\gamma = 2$: (i, CT) as the input, challenger $\mathcal{C}_2$ calls Aut_2 algorithm with SK, then sends $TD_2 = H_5(CT_1{}^{\beta_i}, CT_1, CT_2)$ to $\mathcal{A}_2$.
     (c) $\gamma = 3$: given $(i, CT, j, CT^{'})$ as input, challenger $\mathcal{C}_2$ calls Aut_3 algorithm with SK, then sends $TD_3 = ([H_2(CT_1{}^{\beta_i}, CT_1, CT_2)]_0^{l-1}, (CT_1 CT_1^{'})^{y_i})$ to $\mathcal{A}_2$.
- *Challenge*: Adversary $\mathcal{A}_2$ chooses two plaintext M_0, M_1 and an identity t, there is a contraint that t can not be queried in extract queriy phase or Aut-1 authorization query phase. $\mathcal{C}_2$ picks a bit $b \in \{0, 1\}$ randomly, then encrypts M_b:

$$\begin{aligned} C_{t,1}^* &= g^x, \\ C_{t,2}^* &= M_b \oplus H_2(g^z), \\ C_{t,3}^* &= (x_t \| y_t) \oplus H_5(H_1(t)^{r_t}, C_{t,1}^*, C_{t,2}^*), \end{aligned}$$

  challenger $\mathcal{C}_2$ sends the obtained challenge ciphertext $C^* = (C_{t,1}^*, C_{t,2}^*, C_{t,3}^*)$ to the adversary $\mathcal{A}_2$.
- *Phase*2: $\mathcal{A}_2$ issues the same type query as in Phase 1, and there are two following restrictions:
  1. In the key retrieve query phase or Aut-1 authorizations query phase, i could not be allowed to query;
  2. In the decryption query phase or the authorization query phase, (i, C^*) could not be queried.

- *Guess*: $\mathcal{A}_2$ returns a bit b'. If $b' = b$ holds, it means that $\mathcal{A}_2$ wins the game , then $\mathcal{C}_2$ outputs 1.

□

6. Efficiency Analysis

In Table 1, we describe the communication complexity of our scheme, and compare it with other schemes [4,5,23,24]. $|Z_p|$, $|G|$, $|G_1|$ and $|G_T|$ are used to represent the size of elements in Z_p, G, G_1 and G_T, the second column represents the size of the public key, the third column represents the size of a private key, the four columns represent the size of ciphertext. We can see that our scheme has a smaller size than [4,23,24] in public key and ciphertext, and has a smaller size than [5] in the ciphertext.

Table 1. Communication complexity.

	Public Key	**Secret Key**	**Ciphertext**
PKEET-FA [4]	$3\|G\|$	$3\|Z_p\|$	$5\|G\| + \|Z_p\|$
PKEET-FA [23]	$2\|G\|$	$2\|Z_p\|$	$2\|G\| + 6\|Z_p\|$
PKEET-FA [24]	$2\|G\|$	$2\|Z_p\|$	$\|G\| + 5\|Z_p\|$
IBEET-FA [5]	$\|G_1\|$	$2\|G_1\|$	$5\|G_1\| + \|G_T\| + 2\|Z_p\|$
Our IBEET-FA	$\|G_1\|$	$2\|G_1\|$	$\|G_1\| + 3\|Z_p\|$

In Table 2, we show the comparison of encryption, decryption, authorization, and test in computation complexity. We use "I", "E", and "P" to represent the inversion operation, exponentiation operation and pairing operation, respectively, and represent the comparison of the encryption process, decryption process, authorization process, and test process in computation complexity from the second to fifth columns. In the sixth column, we represent whether the scheme is identity-based, and represents whether the scheme is pairing-based in the last column. Our scheme and [5] have four authorization algorithms. Since Aut-4 is a combination of Aut-1 and Aut-2, we omit Aut-4 for simplicity. In Table 2 and Figure 2, we list the three authorization algorithms of our scheme and [5] for comparison. In the encryption algorithm, Ref. [5] requires seven exponential operations, while our scheme only requires three exponential operations. In the Aut-2 authorization algorithm, Ref. [5] requires one pairing operation, and our scheme only requires two exponential operations. In Aut-3 authorization algorithm, Ref. [5] requires two pairing operations, and our scheme only requires four exponential operations. For the two authorization processes, our scheme reduces the computation costs by 60%, respectively. Reducing the use of pairings is key to reducing computational costs. Compared with [4,23,24], our scheme and [5] are based on identity encryption. The user's public key can be a string related to the user's identity information, which avoids complicated public key certificate management and public key storage. However, Refs. [4,23,24] use public key encryption, which requires a large amount of storage and complex management. Among all the schemes we list, our scheme is the only one that can achieve both ID-based and no pairing.

From the comparison results in Figure 2, it can be seen that the calculation costs of the authorization algorithms of the three authorization methods in our scheme are significantly lower than that of the corresponding three authorization algorithms in Li et al.'s scheme [5]. Compared with other schemes [4,5,23,24], our scheme is more flexible and efficient. In cloud computing, our scheme is applicable to more application scenarios and has high practical significance.

Table 2. Computation complexity.

	Encryption	Decryption	Authorization	Test	ID-Based	Pairings-Based
PKEET-FA [4] Aut-1	6E	5E	0	2P + 2E	NO	YES
PKEET-FA [4] Aut-2	6E	5E	2E	2P + 2E	NO	YES
PKEET-FA [4] Aut-3	6E	5E	2P + 2E	2P + 2E	NO	YES
PKEET-FA [23] Aut-1	4E + 6I	3E + 6I	0	2E + 6I	NO	NO
PKEET-FA [23] Aut-3	4E + 6I	3E + 6I	4E	6E + 6I	NO	NO
PKEET-FA [24] Aut-1	3E + I	2E + I	0	2E + 2I	NO	NO
PKEET-FA [24] Aut-2	3E + I	2E + I	E	2I	NO	NO
PKEET-FA [24] Aut-3	3E + I	2E + I	3E	4E + 4I	NO	NO
IBEET-FA [5] Aut-1	7E	3P + 2E	0	4P	YES	YES
IBEET-FA [5] Aut-2	7E	3P + 2E	P	2P	YES	YES
IBEET-FA [5] Aut-3	7E	3P + 2E	2P	2P	YES	YES
Our IBEET-FA Aut-1	3E	2E	0	2E + 2I	YES	NO
Our IBEET-FA Aut-2	3E	2E	2E	2I	YES	NO
Our IBEET-FA Aut-3	3E	2E	4E	2E + 2I	YES	NO

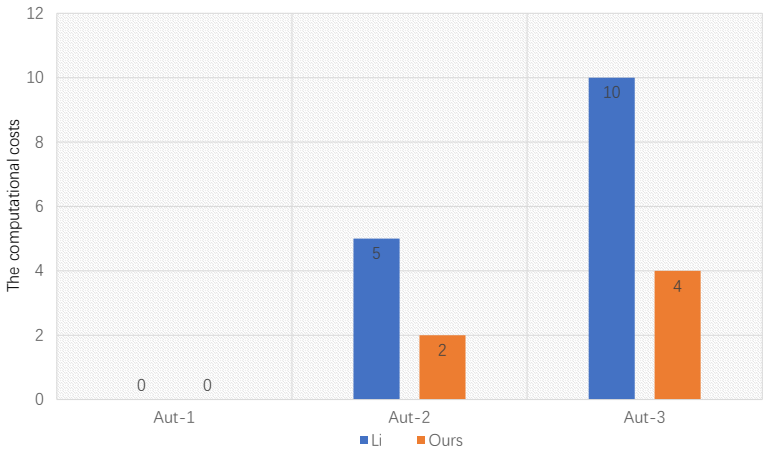

Figure 2. Computational costs comparison of three authorizations with Li [5].

7. Conclusions

In this paper, we propose a new IBE scheme without pairing, which supports the ciphertext equality test. Our scheme introduces the authorization mechanism proposed in the scheme [4], four types of authorization policies providing better flexibility. Compared with works [4,23,24], our scheme is in IBE settings, which means do not need to suffer from complex key store and distribution problems. Compared with works [5], we replaced pairing with discrete logarithms, which helps reduce the computation cost. Specifically, compared to Li et al.'s work, about 57% = (100% − 43%) time cost is saved for the encryption process, and about 60% = (100% − 40%) time costs are saved for the type-2 authorization process and type-3 authorization process. Based on mathematical assumptions, we define the security models of our scheme and prove the security of the scheme.

Our proposed approach can be applied to equality tests over ciphertexts encrypted with different public keys, which increases the application range of cloud computing. Furthermore, our scheme is in IBE settings, avoiding complex key management issues. However, there are security channel key distribution and private key escrow issues in IBE. In the future, we will try to combine the advantages of IBE and PKE to propose more secure and efficient equality test schemes.

Funding: This work was supported by the National Natural Science Foundation of China (NSFC) (No. 61972050), the Beijing Natural Science Foundation (No. L191012) and the 111 Project (No. B08004).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Boneh, D.; Crescenzo, G.D.; Ostrovsky, R.; Persian, G. Public key encryption with keyword search. In Proceedings of the Advances in Cryptology—EUROCRYPT 2004, Interlaken, Switzerland, 2–6 May 2004.
2. Yang, G.; Tan, C.H.; Huang, Q.; Wong, D.S. Probabilistic public key encryption with equality test. In Proceedings of the Topics in Cryptology—CT-RSA 2010, San Francisco, CA, USA, 1–5 March 2010.
3. Tang, Q. Towards public key encryption scheme supporting equality test with fine grained authorization. In Proceedings of the Australisian Conference on Information Security and Privacy(ACISP), Melbourne, Australia, 11–13 July 2011.
4. Ma, S.; Huang, Q.; Zhang, M.W.; Yang, B. Efficient public key encryption with equality test supporting flexible authorization. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 458–470. [CrossRef]
5. Li, H.B.; Huang, Q.; Ma, S.; Shen, J.; Susilo, W. Authorized equality test on identity-based ciphertexts for secret data sharing via cloud storage. *IEEE Access* **2019**, *7*, 25409–25421. [CrossRef]
6. Shamir, A. Identity-based cryptosystems and signature schemes. In Proceedings of the Advances in Cryptology—CRYPTO 1984, Santa Barbara, CA, USA, 19–22 August 1984.
7. Paterson, K.G.; Srinivasan, S. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Des. Codes Cryptogr.* **2009**, *52*, 219–241. [CrossRef]
8. Chuah, M.; Hu, W. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In Proceedings of the 31st International Conference on Distributed Computing Systems Workshops, Minneapolis, MN, USA, 20–24 June 2011.
9. Park, D.J.; Kim, K.; Lee, P.J. Public key encryption with conjunctive field keyword search. In Proceedings of the International Conference on Information Security Applications(WISA), Jeju Island, Republic of Korea, 23–25 August 2004.
10. Byun, J.W.; Rhee, H.S.; Park, H.A.; Lee, D.H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In Proceedings of the Secure Data Management(SDM), Seoul, Republic of Korea, 10–11 September 2006.
11. Bellare, M.; Boldyreva, A.; O'Neill, A. Deterministic and efficiently searchable encryption. In Proceedings of the Advances in Cryptology—CRYPTO 2007, Santa Barbara, CA, USA, 19–23 August 2007.
12. Tang, Q. Public key encryption schemes supporting equality test with authorisation of different granularity. *IJACT* **2012**, *2*, 304–321. [CrossRef]
13. Tang, Q. Public key encryption supporting plaintext equality test and user-specified authorization. *Secur. Commun. Netw.* **2012**, *5*, 1351–1362. [CrossRef]
14. Ma, S.; Zhang, M.W; Huang, Q.; Yang, B. Public Key Encryption with Delegated Equality Test in a Multi-User Setting. *Comput. J.* **2015**, *58*, 986–1002. [CrossRef]
15. Wu, T.; Ma, S.; Mu, Y.; Zeng, S.K. ID-Based Encryption with Equality Test Against Insider Attack. In Proceedings of the Australasian Conference on Information Security and Privacy(ACISP), Auckland, New Zealand, 3–5 July 2017.
16. Ma, S. Authorized Equality Test of Encrypted Data for Secure Cloud Databases. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security Furthermore, Privacy in Computing Furthermore, Communications/12th IEEE International Conference On Big Data Science Furthermore, Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018.
17. Huang, K.B.; Raylin, T.; Yu-Chi, C. PKE-AET:Public Key Encryption with Authorized Equality Test. *Comput. J.* **2015**, *58*, 2686–2697. [CrossRef]
18. Huang, K.B.; Raylin, T.; Yu-Chi, C. A New Public Key Encryption with Equality Test. In Proceedings of the Network and System Security (NSS), Xi'an, China, 15–17 October 2014.
19. Huang, K.B.; Yu-Chi, C.; Raylin, T. Semantic Secure Public Key Encryption with Filtered Equality Test-PKE-FET. In Proceedings of the 12th International Joint Conference on e-Business and Telecommunications (ICETE), Colmar, France, 20–22 July 2015.
20. Huang, K.B.; Raylin, T.; Yu-Chi, C. Somewhat semantic secure public key encryption with filtered-equality-test in the standard model and its extension to searchable encryption. *J. Comput. Syst. Sci.* **2017**, *89*, 400–409. [CrossRef]
21. Zhu, H.J.; Wang, L.C.; Ahmad, H.; Niu, X.X. Key-policy attribute-based encryption with equality test in cloud computing. *IEEE Access* **2017**, *5*, 20428–20439. [CrossRef]
22. Wang, Q.; Peng, L.; Xiong, H.; Sun, J.F.; Qin, Z.G. Ciphertext-policy attribute-based encryption with delegated equality test in cloud computing. *IEEE Access* **2018**, *6*, 760–771. [CrossRef]
23. Lin, X.J.; Sun, L.; Qu, H.p.; Zhang, X.S. Public key encryption supporting equality test and flexible authorization without bilinear pairings. *Comput. Commun.* **2021**, *170*, 190–199. [CrossRef]
24. Zhu, H.J.; Wang, L.C.; Ahmad, H.; Niu, X.X. Pairing-free equality test over short ciphertexts. *Int. J. Distrib. Sens. Netw.* **2017**, *13*. [CrossRef]
25. Ma, S. Identity-based encryption with outsourced equality test in cloud computing. *Inf. Sci.* **2016**, *328*, 389–402. [CrossRef]
26. Duong, D.H.; Roy, P.S.; Susilo, W.; Fukushima, K.; Kiyomoto, S.; Sipasseuth, A. Chosen-ciphertext lattice-based public key encryption with equality test in standard model. *Theor. Comput. Sci.* **2022**, *905*, 31–53. [CrossRef]
27. Amerini, I.; Anagnostopoulos, A.; Maiano, L.; Celsi, L.R. Deep Learning for Multimedia Forensics. *Found. Trends Comput. Graph. Vis.* **2021**, *12*, 309–457. [CrossRef]
28. Susilo, W.; Guo, F.C.; Zhao, Z.; Wu, G. PKE-MET: Public-key encryption With multi-ciphertext equality test in cloud computing. *IEEE Trans. Cloud Comput.* **2022**, *10*, 1476–1488. [CrossRef]

29. Lin, X.J.; Wang, Q.; Sun, L.; Qu, H. Identity-based encryption with equality test and datestamp-based authorization mechanism. *Theor. Comput. Sci.* **2021**, *861*, 117–132. [CrossRef]
30. Deverajan, G.G.; Muthukumaran, V.; Hsu, C.; Karuppiah, M.; Chung, Y.; Chen, Y. Public key encryption with equality test for Industrial Internet of Things system in cloud computing. *Trans. Emerg. Telecommun. Technol.* **2021**, *33*, e4202. [CrossRef]
31. Vaanchig, N.; Qin, Z.; Ragchaasuren, B. Constructing secure-channel free identity-based encryption with equality test for vehicle-data sharing in cloud computing. *Trans. Emerg. Telecommun. Technol.* **2020**, *33*, e3896. [CrossRef]