Certificateless Public Key Encryption with Equality Test

Xi-Jun Lin [*], Zhen Yan [†], Qi Zhang [‡] and Haipeng Qu [§]

December 2, 2016

Abstract

In this paper, we propose the concept of certificateless public key encryption with equality test (CL-PKEET) to solve the key escrow problem in IBEET. More in details, we first give the definition of CL-PKEET and define the security model. Finally, we propose a concrete CL-PKEET scheme based on the Decision Bilinear Diffie-Hellman (DBDH) assumption and prove its security.

KeyWords: certificateless public key encryption; public key encryption; equality test; authorition

1 Introduction

In the cloud era, plenty of cloud services offer a broad set of global compute, storage, database, analytics, application, and deployment services that help organizations move faster, lower IT costs, and scale applications. Considering the potential privacy risks, cryptosystems can be used to encrypt the private data, which are stored in the cloud server. Meanwhile, in order to maintain the database, we need a solution to support computations on the encrypted data. Public key encryption with keyword search (PEKS) [2, 5] is a well-known solution, which supports keyword searching over ciphertexts without retrieving messages by using the corresponding trapdoors [4]. However, the ciphertexts in PEKS are encrypted under the same public key, it does not apply to many computations on cloud.

Yang et al. proposed the public key encryption with equality test (PKEET) [15], which can be performed on ciphertexts with different public keys. Equality test can be defined like that: Let C, C' be two ciphertexts encrypted under two different public keys, where $C = \mathit{Encrypt}(M, PK)$ and $C' = \mathit{Encrypt}(M', PK')$, this algorithm checks whether or not $M = M'$ holds. If it is the case, it returns 1, and 0 otherwise. In a certain sense, PKEET, which trivially supports the traditional functionality of PEKS, is an extension of PEKS. However, the adversary is able to check the equality of ciphertexts without any authorization, which violates the data owners' privacy. Hence, several primitives with authorization mechanism were proposed.

1.1 Related Work

The concept of public key encryption with equality test (PKEET) [15] was first proposed by Yang et al. and their cryptosystem can be used to check whether two ciphertexts are encryptions of the same message in outsourced database.

Tang integrated a fine-grained authorization policy enforcement mechanism into PKEET and proposed an enhanced primitive, called FG-PKEET [13]. Tang proposed AoN-PKEET [12], which introduces an authorization mechanism for users to specify who can independently perform a plaintext equality test from their ciphertexts. To mitigate the inherent offline message recovery attacks, Tang extended FG-PKEET to a two-proxy setting [14], where two proxies need to collaborate in order to perform an equality test.

[*]X.J.Lin is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China.

[†]Z.Yan is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China.

[‡]Q.Zhang is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China.

[§]H.Qu, corresponding author, is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China.

Ma et al. introduced the notion of public key encryption with delegated equality test (PKE-DET) [8], which requires only the delegated party to deal with the work in a practical multi-user setting. Huang et al. proposed public key encryption scheme with authorized equality test(PKE-AET) [6] which provides two kinds of warrants that are referred to as receiver's warrants and cipher-warrants. And in order to strengthen the privacy protection, Ma et al. proposed a new primitive, called public key encryption with equality test supporting flexible authorization (PKEET-FA) [10] which supports four different types of authorization.

1.2 Our Contribution

Recently, Ma proposed identity-based encryption with equality test (IBEET) [9] by combining the concepts of public key encryption with equality test and identity-based public key cryptography (ID-PKC) [11].

ID-PKC eliminates the need for certificates and some of the problems associated with them, but its dependence on the private key generator(PKG) introduces key escrow problem for using a master key to generate private keys. Obviously, IBEET suffers the key escrow problem as well. Fortunately, Al-Riyami et al. [1] proposed certificateless public key cryptography (CL-PKC), which does not require the use of certificates and yet does not have the key escrow problem in ID-PKC.

We believe that a primitive which has the features of CL-PKC and PKEET could enjoy both convenience and security since the certificate management problem and the key escrow problem could be solved simultaneously. Motivated by this, we propose a new primitive, called certificateless public key encryption with equality test (CL-PKEET). Our contribution can be summarized as follows.

1. We propose the concept of CL-PKEET and define the security models where four different types of adversaries are considered.

2. We propose a concrete CL-PKEET scheme based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption, and then prove its security.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we recall the definition of asymmetric bilinear group and decisional bilinear Diffie-Hellman (DBDH) assumption. In Section 3, we introduce the definition of CL-PKEET scheme and its security models. Then we present a concrete CL-PKEET scheme in Section 4 and its security is proven in Section 5. In Section 6, we conclude our work.

2 Preliminaries

2.1 Asymmetric Bilinear Groups

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicative cyclic groups of prime order q with the security parameter λ. Let g_1, g_2 be generators of $\mathbb{G}_1$, $\mathbb{G}_2$ respectively. The asymmetric bilinear map e:$\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ has the following properties:

- Bilinearity: $\forall u \in \mathbb{G}_1$, $\forall v \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}_q^*$, $e(u^a, v^b) = e(u, v)^{ab}$

- Non-degeneracy: $\exists g_1 \in \mathbb{G}_1$,$g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2)$ has order q, that is, $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

- Computability:$\forall (u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$, $e(u, v)$ is efficiently computable.

We say that $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are bilinear groups with no efficiently computable isomorphisms if the group operations in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable, but there are no effiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ [3, 7].

2.2 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Let $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a description of the asymmetric bilinear groups of prime order q with the security parameter λ defined above. Let g_1,g_2 be generators of $\mathbb{G}_1$, $\mathbb{G}_2$ respectively. The assumption is that if the challenge values

$$D = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$$

and h are given, no polynomial time algorithm $\mathcal{A}$ can distinguish $e(g_1, g_2)^{abc}$ from $h \in \mathbb{G}_T$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as

$$Adv_{DBDH}^{\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, e(g_1, g_2)^{abc}) = 1] - Pr[\mathcal{A}(D, h) = 1]|$$

where the probability is taken over the random choice of $a, b, c \in \mathbb{Z}_q^*$ and $h \in \mathbb{G}_T$ [3].

3 Definition of CL-PKEET

In this section, we first provide the framework of CL-PKEET, and then present the formal definition and the security models.

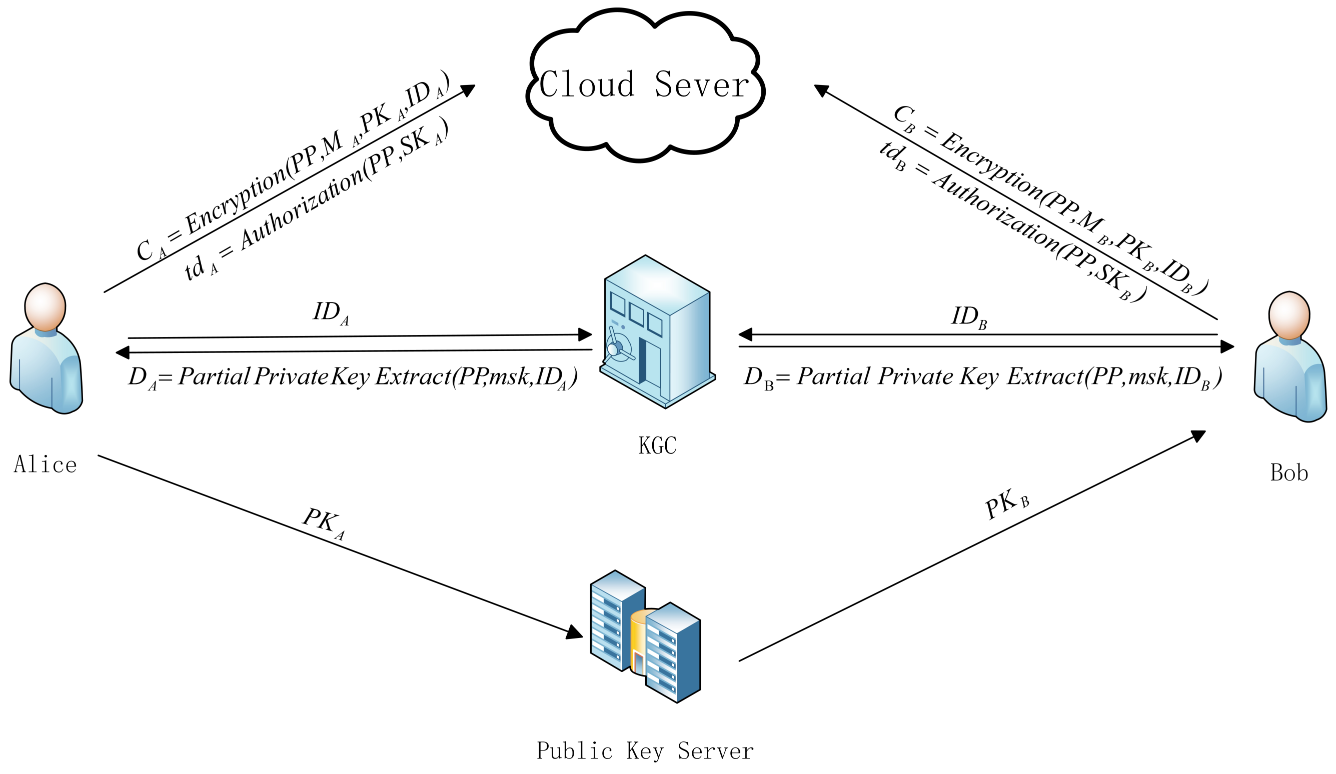
Cloud Sever
$C_A = Encryption(PP, M_A, PK_A, ID_A)$
$td_A = Authorization(PP, SK_A)$
$C_B = Encryption(PP, M_B, PK_B, ID_B)$
$td_B = Authorization(PP, SK_B)$
ID_A
$D_A = Partial\ Private\ Key\ Extract(PP, msk, ID_A)$
ID_B
$D_B = Partial\ Private\ Key\ Extract(PP, msk, ID_B)$
Alice
KGC
Bob
PK_A
PK_B
Public Key Server

Figure 1: CL-PKEET system model

As illustrated in Figure 1, CL-PKEET consists of four types of entitles: users, the key generation center (KGC), the cloud server and the public key server. All ciphertexts sent to the users are stored in the cloud server. If the users (e.g. Alice and Bob) want to compare their ciphertexts, they can generate and send the corresponding trapdoors to the cloud server. As a result, the cloud server can compare two ciphertexts with the trapdoors. Note that the KGC's task is to generate and distribute partial private keys for the users secretly. The cloud server's task is to store and compare the ciphertexts. The public key server's task is to store the public keys of the users. More in details, the work flow of our proposal is presented as follow:

1. The KGC generates the partial private key with respect to the user's identity and sends it to the user secretly.

2. Upon receiving the partial private key from the KGC, the user generates its (full) private key and keeps it by itself. Moreover, the user generates its public key and stores it into the public key server.

3. To compare its ciphertexts with the other users' ciphertexts, the user (e.g. Alice or Bob) generates the trapdoor with its private key which is sent to the cloud server.

4. With the trapdoors from the users, the cloud server can check whether two ciphertexts are generated on the same message.

We stress here that it is assumed that the KGC and the servers are honest but may be curious. Moreover, the KGC and the servers do not collude with each other.

3.1 Definition

Let $\mathbb{M}$ be the message space and $\mathbb{C}$ be the ciphertext space. The definition of the CL-PKEET scheme is as follows:

- *Setup*(λ): This algorithm,run by the KGC, takes a security parameter λ as input, and outputs the system public parameter PP and master key msk.

- *Partial Private Key Extract*(PP, msk, ID): This algorithm, run by the KGC, takes the public parameter PP, the master key msk and a user's identity $ID \in \{0,1\}^*$as input. It returns the user's partial private key D.

- *Set Secret Value*(PP): This algorithm, run by a user, takes the public parameter PP as input, and outputs the user's secret value x.

- *Set Private Key*(PP, D, x): This algorithm, run by a user, takes the public parameter PP, the user's partial private key D and secret value x as input. It returns the user's full private key SK.

- *Set Public Key*(PP, x): This algorithm, run by a user, takes public parameter PP and the user's secret value x as input, and outputs the user's public key PK .

- *Encryption*(PP, M, PK, ID): This algorithm, run by a message sender, takes the public parameter PP, a message $M \in \mathbb{M}$, a user's public key PK and identity ID as input. It returns either a ciphertext $C \in \mathbb{C}$ or a null symbol $\perp$ indicating an encryption failure.

- *Decryption*(PP, C, SK): This algorithm, run by a user, takes the public parameter PP, a ciphertext $C \in \mathbb{C}$ and the user's private key SK as input. It returns the corresponding message $M \in \mathbb{M}$ or a null symbol $\perp$ indicating a decryption failure.

- *Authorization*(PP, SK): This algorithm, run by a user, takes the public parameter PP and the user's private key SK as input, and returns the user's trapdoor td.

- *Test*$(PP, C_A, td_A, C_B, td_B)$: Let C_A and td_A be the user A's ciphertext and trapdoor, and C_B and td_B be the user B's ciphertext and trapdoor. This algorithm, run by the cloud server, takes the public parameter PP, a ciphertext $C_A \in \mathbb{C}$, a trapdoor td_A, a ciphertext $C_B \in \mathbb{C}$ and a trapdoor td_B as input, and returns 1 if C_A and C_B contain the same message; or 0 otherwise.

For the consistency property, these algorithms must satisfy the following conditions:

1. $\mathcal{M} = \textit{Decryption}(PP, \textit{Encryption}(PP, M, PK, ID), SK)$, where SK and PK are private key and public key with respect to the identity ID respectively.

2. Let $C_A = \textit{Encryption}(PP, M, PK_A, ID_A)$, $C_B = \textit{Encryption}(PP, M, PK_B, ID_B)$, $td_A = \textit{Authorization}$ (PP, SK_A) and $td_B = \textit{Authorization}(PP, SK_B)$, where SK_A and PK_A are private key and public key with respect to the identity ID_A, SK_B and PK_B are private key and public key with respect to the identity ID_B. If $M_A = M_B$, $\textit{Test}(C_A, td_A, C_B, td_B) = 1$; otherwise, $\Pr[\textit{Test}(C_A, td_A, C_B, td_B) = 1]$ is negligible.

3.2 Security Models

By the security models of PKEET and CL-PKE, there are four types of adversaries defined for the security of CL-PKEET:

- *Type*-1 *adversary:* This type of adversary does not have access to the master key, but may replace public keys with values of its choice. Moreover, without the trapdoor, it cannot decide whether the challenge ciphertext is the encryption of which message. With respect to such an adversary, we define the notion of IND-CCA security.
- *Type*-2 *adversary:* This type of adversary does have access to the master key, but may not replace the public key of any user. Moreover, without the trapdoor, it cannot decide whether the challenge ciphertext is the encryption of which message. With respect to such an adversary, we define the notion of IND-CCA security.
- *Type*-3 *adversary:* This type of adversary does not have access to the master key, but may replace public keys with values of its choice. Moreover, with the trapdoor, it cannot retrieve the message from the challenge ciphertext. With respect to such an adversary, we define the notion of OW-CCA security.
- *Type*-4 *adversary:* This type of adversary does have access to the master key, but may not replace the public key of any user. Moreover, with the trapdoor, it cannot retrieve the message from the challenge ciphertext. With respect to such an adversary, we define the notion of OW-CCA security.

3.2.1 IND-CCA Security for Type-1 Adversary

The formal definition of IND-CCA security for authorization against Type-1 adversary is defined below:

Game 1 *Let $\mathcal{A}_1$ be a Type-*1 *adversary. The game between the challenger $\mathcal{C}$ and $\mathcal{A}_1$ is as follows:*

1. *Setup:* $\mathcal{C}$ runs the algorithm *Setup* to generate the public parameter PP and the master key msk, and then PP are given to $\mathcal{A}_1$, meanwhile, msk is kept by $\mathcal{C}$ itself.
2. *Phase 1:* The following queries can be issued by $\mathcal{A}_1$ for polynomially many times, and restricted by the rule of adversary behavior.
   - *Partial private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding partial private key D_i.
   - *Private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding private key SK_i.
   - *Public key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding public key PK_i.
   - *Replace public key*(ID_i,PK_i'): Upon receiving a user's identity ID_i and a public key PK_i', $\mathcal{C}$ replaces the corresponding public key of the user with PK_i'.
   - *Decryption query*(ID_i,C): Upon receiving a user's identity ID_i and a ciphertext C, $\mathcal{C}$ responds with the output of the algorithm *Decryption*(PP, C, SK_i), where SK_i is the private key with respect to the identity ID_i.
   - *Authorization query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding trapdoor td_i.
3. *Challenge:* $\mathcal{A}_1$ submits two equal-length messages M_0^*,M_1^* and an identity ID^*. $\mathcal{C}$ picks $\rho \in \{0,1\}$ randomly, and then computes the challenge ciphertext $C^* = Encryption(PP, M_\rho^*, PK^*, ID^*)$, where PK^* is the current public key with respect to the identity ID^*. If the encryption outputs $\perp$, $\mathcal{A}_1$ immediately loses the game. Otherwise, $\mathcal{C}$ returns C^* to $\mathcal{A}_1$.
4. *Phase 2:* $\mathcal{A}_1$ issues queries as done in *Phase* 1.
5. *Guess:* $\mathcal{A}_1$ outputs $\rho' \in \{0,1\}$, and wins the game if $\rho = \rho'$. The advantage of $\mathcal{A}_1$ is defined as $Adv_{CL-PKEET,\mathcal{A}_1}^{IND-CCA,Type-1}(\lambda) = |\Pr[\rho = \rho'] - \frac{1}{2}|$, where λ is the security parameter.

There are some constraints on $\mathcal{A}_1$ as follows:

- ID^* does not appear in the *Private key query* at any point.
- If ID^*'s public key has been replaced, ID^* does not appear in the *Partial private key query.*
- If a user's public key has been replaced, the corresponding identity ID does not appear in the *Private key query.*
- (ID^*, C^*) does not appear in the *Decryption query.*
- ID^* does not appear in the *Authorization query.*

Definition 1 *We say that CL-PKEET has IND-CCA security for authorization if for any PPT adversaries $\mathcal{A}_1$, its advantage $Adv^{IND-CCA,Type-1}_{CL-PKEET,\mathcal{A}_1}(\lambda)$ is negligible in the security parameter λ.*

3.2.2 IND-CCA Security for Type-2 Adversary

The formal definition of IND-CCA security for authorization against Type-2 adversary is defined below:

Game 2 *Let $\mathcal{A}_2$ be a Type-2 adversary. The game between the challenger $\mathcal{C}$ and $\mathcal{A}_2$ is as follows:*

1. *Setup:* $\mathcal{C}$ runs the algorithm *Setup* to generate the public parameter PP and the master key msk which are given to $\mathcal{A}_2$.
2. *Phase 1:* The following queries can be issued by $\mathcal{A}_2$ for polynomially many times, and restricted by the rule of adversary behavior.
    - *Private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding private key SK_i.
    - *Public key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding public key PK_i.
    - *Decryption query*(ID_i,C): Upon receiving a user's identity ID_i and a ciphertext C, $\mathcal{C}$ responds with the output of the algorithm *Decryption*(PP, C, SK_i), where SK_i is the private key with respect to the identity ID_i.
    - *Authorization query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding trapdoor td_i.
3. *Challenge:* $\mathcal{A}_2$ submits two equal-length messages M_0^*,M_1^* and an identity ID^*. $\mathcal{C}$ randomly picks $\rho \in \{0, 1\}$, and then returns the challenge ciphertext $C^* = Encryption(PP, M_\rho^*, PK^*, ID^*)$, where PK^* is the public key with respect to the identity ID^*.If the encryption outputs $\perp$, $\mathcal{A}_2$ immediately loses the game. Otherwise, $\mathcal{C}$ returns C^* to $\mathcal{A}_2$.
4. *Phase 2:* $\mathcal{A}_2$ issues queries as done in *Phase* 1.
5. *Guess:* $\mathcal{A}_2$ outputs $\rho' \in \{0, 1\}$, and wins the game if $\rho = \rho'$. The advantage of $\mathcal{A}_2$ is defined as $Adv^{IND-CCA,Type-2}_{CL-PKEET,\mathcal{A}_2}(\lambda) = |\Pr[\rho = \rho'] - \frac{1}{2}|$, where λ is the security parameter.

There are some constraints on $\mathcal{A}_2$ as follows:

- ID^* does not appear in the *Private key query* at any point.
- (ID^*, C^*) does not appear in the *Decryption query.*
- ID^* does not appear in the *Authorization query.*

Definition 2 *We say that CL-PKEET has IND-CCA security for authorization if for any PPT adversaries $\mathcal{A}_2$, its advantage $Adv^{IND-CCA,Type-2}_{CL-PKEET,\mathcal{A}_2}(\lambda)$ is negligible in the security parameter λ.*

3.2.3 OW-CCA Security for Type-3 Adversary

The formal definition of OW-CCA security for authorization against Type-3 adversary is defined below:

Game 3 *Let $\mathcal{A}_3$ be a Type-3 adversary. The game between the challenger $\mathcal{C}$ and $\mathcal{A}_3$ is as follows:*

1. *Setup:* $\mathcal{C}$ runs the algorithm *Setup* to generate the public parameter PP and the master key msk, and then PP are given to $\mathcal{A}_3$, meanwhile, msk is kept by $\mathcal{C}$ itself.

2. *Phase 1:* The following queries can be issued by $\mathcal{A}_3$ for polynomially many times, and restricted by the rule of adversary behavior.

    - *Partial private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding partial private key D_i.
    - *Private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding private key SK_i.
    - *Public key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding public key PK_i.
    - *Replace public key*(ID_i,PK_i'): Upon receiving a user's identity ID_i and a public key PK_i', $\mathcal{C}$ replaces the corresponding public key of the user with PK_i'.
    - *Decryption query*(ID_i,C): Upon receiving a user's identity ID_i and a ciphertext C, $\mathcal{C}$ responds with the output of the algorithm $Decryption(PP, C, SK_i)$, where SK_i is the private key with respect to the identity ID_i.
    - *Authorization query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding trapdoor td_i.

3. *Challenge:* $\mathcal{C}$ picks a message M^* and an identity ID^* randomly, and then returns the challenge ciphertext $C^* = Encryption(PP, M^*, PK^*, ID^*)$, where PK^* is the current public key with respect to the identity ID^*.

4. *Phase 2:* $\mathcal{A}_3$ issues queries as done in *Phase* 1.

5. *Guess:* $\mathcal{A}_3$ outputs $M^{'}$, and wins the game if $M^* = M^{'}$. The advantage of $\mathcal{A}_3$ is defined as $Adv_{CL-PKEET,\mathcal{A}_3}^{OW-CCA,Type-3}(\lambda) = Pr[M^* = M^{'}]$, where λ is the security parameter.

There are some constraints on $\mathcal{A}_3$ as follows:

- ID^* does not appear in the *Private key query* at any point.
- If ID^*'s public key has been replaced, ID^* does not appear in the *Partial private key query*.
- If a user's public key has been replaced, the corresponding identity ID does not appear in the *Private key query*.
- (ID^*, C^*) does not appear in the *Decryption query*.

Definition 3 *We say that CL-PKEET has OW-CCA security for authorization if for any PPT adversaries $\mathcal{A}_3$, its advantage $Adv_{CL-PKEET,\mathcal{A}_3}^{OW-CCA,Type-3}(\lambda)$ is negligible in the security parameter λ.*

3.2.4 OW-CCA Security for Type-4 Adversary

The formal definition of OW-CCA security for authorization against Type-4 adversary is defined below:

Game 4 *Let $\mathcal{A}_4$ be a Type-4 adversary. The game between the challenger $\mathcal{C}$ and $\mathcal{A}_4$ is as follows:*

1. *Setup:* $\mathcal{C}$ runs the algorithm *Setup* to generate the public parameter PP and the master key msk which are given to $\mathcal{A}_4$.

2. *Phase 1:* The following queries can be issued by $\mathcal{A}_4$ for polynomially many times, and restricted by the rule of adversary behavior.

- *Private key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding private key SK_i.
- *Public key query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding public key PK_i.
- *Decryption query*(ID_i, C): Upon receiving a user's identity ID_i and a ciphertext C, $\mathcal{C}$ responds with the output of the algorithm $Decryption(PP, C, SK_i)$, where SK_i is the private key with respect to the identity ID_i.
- *Authorization query*(ID_i): Upon receiving a user's identity ID_i, $\mathcal{C}$ responds with the corresponding trapdoor td_i.

3. *Challenge:* $\mathcal{C}$ picks a message M^* and an identity ID^* randomly, and then returns the challenge ciphertext $C^* = Encryption(PP, M^*, PK^*, ID^*)$, where PK^* is the public key with respect to the identity ID^*.
4. *Phase 2:* $\mathcal{A}_4$ issues queries as done in *Phase* 1.
5. *Guess:* $\mathcal{A}_4$ outputs $M^{'}$, and wins the game if $M^* = M^{'}$. The advantage of $\mathcal{A}_4$ is defined as $Adv_{CL-PKEET,\mathcal{A}_4}^{OW-CCA,Type-4}(\lambda) = Pr[M^* = M^{'}]$, where λ is the security parameter.

There are some constraints on $\mathcal{A}_4$ as follows:

- ID^* does not appear in the *Private key query* at any point.
- (ID^*, C^*) does not appear in the *Decryption query*.

Definition 4 *We say that CL-PKEET has OW-CCA security for authorization if for any PPT adversaries $\mathcal{A}_4$, its advantage $Adv_{CL-PKEET,\mathcal{A}_4}^{OW-CCA,Type-4}(\lambda)$ is negligible in the security parameter λ.*

4 The Proposed CL-PKEET Scheme

In this section, we propose a concrete CL-PKEET scheme, which consists of the following algorithms:

- *Setup*(λ): This algorithm performs as follows:
  1. Generate a asymmetric bilinear group $\mathcal{G} := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$.
  2. Pick a random generator $g_1 \in \mathbb{G}_1$, a random generator $g_2 \in \mathbb{G}_2$, and a random master key $s \in \mathbb{Z}_q^*$.
  3. Let $\hat{g_1} = g_1^s$ and $\hat{g_2} = g_2^s$.
  4. Select cryptographic secure hash functions: $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \to \mathbb{G}_2$, $H_3 : \mathbb{G}_T \times \mathbb{G}_1^2 \times \mathbb{G}_2 \to \{0,1\}^{n+l}$, $H_4 : \{0,1\}^n \to \mathbb{G}_2$, $H_5 : \{0,1\}^{n+l} \to \mathbb{Z}_q^*$ and $H_6 : \mathbb{G}_T \to \mathbb{G}_2$.
  5. Output the public parameter
  $$PP = (\mathcal{G}, g_1, g_2, \hat{g_1}, \hat{g_2}, H_1, H_2, H_3, H_4, H_5, H_6)$$
- *Partial Private Key Extract* (PP, s, ID): This algorithm returns the user's partial private key $D = (D_1, D_2)$, where $D_1 = H_1(ID)^s$ and $D_2 = H_2(ID)^s$.
- *Set Secret Value*(PP): This algorithm outputs a random number $x \in \mathbb{Z}_q^*$ as the secret value.
- *Set Private Key*(PP, D, x): Let $D = (D_1, D_2)$. This algorithm returns the user's private key $SK = (S_1, S_2)$, where $S_1 = {D_1}^x$ and $S_2 = {D_2}^x$.
- *Set Public Key*(PP, x): This algorithm outputs the user's public key $PK = (X, Y)$, where $X = \hat{g_2}^x$ and $Y = \hat{g_1}^x$.
- *Encryption*(PP, M, PK, ID): Let $M \in \{0,1\}^n$ and $PK = (X, Y)$. This algorithm generates a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$ as follows:
  1. Pick three random numbers $\alpha, \beta \in \mathbb{Z}_q^*$ and $k \in \{0,1\}^l$, then compute $R = H_5(M, k)$.

2. Compute the ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$ as follows
$$\begin{cases} C_1 = g_1^R, \\ C_2 = g_1^\alpha, \\ C_3 = g_2^\beta, \\ C_4 = H_3(e(Y^\alpha, Q_2), C_1, C_2, C_3) \oplus (M \parallel k), \\ C_5 = H_4(M)^R \cdot H_6(e(Q_1, X^\beta)), \end{cases}$$
where $Q_1 = H_1(ID)$ and $Q_2 = H_2(ID)$.

- *Decryption*(PP, C, SK): Let $C = (C_1, C_2, C_3, C_4, C_5)$ and $SK = (S_1, S_2)$. This algorithm performs as follows:
  1. Recover $M' \parallel k'$ by computing $C_4 \oplus H_3(e(C_2, S_2), C_1, C_2, C_3)$.
  2. Recover R' by computing $H_5(M', k')$.
  3. If $C_1 = g_1^{R'}$ and $\frac{C_5}{H_4(M')^{R'}} = H_6(e(S_1, C_3))$ both hold, output M'; otherwise, output $\perp$.
- *Authorization*(PP, SK): Let $SK = (S_1, S_2)$. This algorithm outputs the trapdoor $td = S_1$.
- *Test$(PP, C_A, td_A, C_B, td_B)$:* Let $C_A = (C_{A,1}, C_{A,2}, C_{A,3}, C_{A,4}, C_{A,5})$ and $C_B = (C_{B,1}, C_{B,2}, C_{B,3}, C_{B,4}, C_{B,5})$. This algorithm computes
$$K_A = \frac{C_{A,5}}{H_6(e(td_A, C_{A,3}))}$$
$$K_B = \frac{C_{B,5}}{H_6(e(td_B, C_{B,3}))}$$
and then checks whether $e(C_{A,1}, K_B) = e(C_{B,1}, K_A)$ holds. If it is the case, it returns 1; or 0 otherwise.

The above CL-PKEET scheme satisfies the consistency property which is proven as follows:

1. As to the first condition, it is straightforward to be verified:
$$\begin{aligned} & C_4 \oplus H_3(e(C_2, S_2), C_1, C_2, C_3) \\ & = H_3(e(Y^\alpha, Q_2), C_1, C_2, C_3) \oplus (M \parallel k) \oplus H_3(e(C_2, S_2), C_1, C_2, C_3) \\ & = H_3(e(\hat{g_1}^x, Q_2)^\alpha, C_1, C_2, C_3) \oplus (M \parallel k) \oplus H_3(e(g_1^\alpha, D_2^x), C_1, C_2, C_3) \\ & = H_3(e(g_1^{sx}, Q_2)^\alpha, C_1, C_2, C_3) \oplus (M \parallel k) \oplus H_3(e(g_1^\alpha, Q_2^{sx}), C_1, C_2, C_3) \\ & = M \parallel k \end{aligned}$$
2. As to the second condition, assuming the ciphertexts are well-formed for ID_A and ID_B
$$K_A = \frac{C_{A,5}}{H_6(e(td_A, C_{A,3}))} = \frac{H_4(M_A)^{R_A} \cdot H_6(e(H_1(ID_A), X_A^{\beta_A}))}{H_6(e(td_A, C_{A,3}))} = \frac{H_4(M_A)^{R_A} \cdot H_6(e(H_1(ID_A), g_2^{sx_A \beta_A}))}{H_6(e(H_1(ID_A)^{sx_A}, g_2^{\beta_A}))} = H_4(M_A)^{R_A}$$
$$K_B = \frac{C_{B,5}}{H_6(e(td_B, C_{B,3}))} = \frac{H_4(M_B)^{R_B} \cdot H_6(e(H_1(ID_B), X_B^{\beta_B}))}{H_6(e(td_B, C_{B,3}))} = \frac{H_4(M_B)^{R_B} \cdot H_6(e(H_1(ID_B), g_2^{sx_B \beta_B}))}{H_6(e(H_1(ID_B)^{sx_B}, g_2^{\beta_B}))} = H_4(M_B)^{R_B}$$
$$e(C_{A,1}, K_B) = e(g_1^{R_A}, H_4(M_B)^{R_B}) = e(g_1, H_4(M_B)^{R_A R_B})$$
$$e(C_{B,1}, K_A) = e(g_1^{R_B}, H_4(M_A)^{R_A}) = e(g_1, H_4(M_A)^{R_A R_B})$$

If $M_A = M_B$, then $e(C_{A,1}, K_B) = e(C_{B,1}, K_A)$ holds, so $Test(C_A, td_A, C_B, td_B)$ outputs 1; or 0 otherwise.

5 Security Analysis

Theorem 1 *Our proposed CL-PKEET scheme is IND-CCA secure for the authorization against Type-1 adversary (c.f. Definition 1) based on DBDH assumption in the random oracle model.*

Proof: Assume that $\mathcal{A}_1$ (a Type-1 adversary) can break our proposal, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ to solve the DBDH problem. Let $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, h)$ be an instance of the DBDH problem, $\mathcal{B}$'s task is to decide whether or not $e(g_1, g_2)^{abc} = h$ holds. $\mathcal{B}$ and $\mathcal{A}_1$ play the following game.

1. *Setup:* $\mathcal{B}$ generates the public parameter $PP = (\mathcal{G}, g_1, g_2, \hat{g_1}, \hat{g_2}, H_1, H_2, H_3, H_4, H_5, H_6)$, where $\hat{g_1} = g_1^a$ and $\hat{g_2} = g_2^a$. Then $\mathcal{B}$ sends PP to $\mathcal{A}_1$, and picks an index $I(1 \leq I \leq q_{H_2})$ randomly as the target identity index, where $q_{H_2} \geq 0$ is the numbers of distinct H_2-*query* made by $\mathcal{A}_1$. Lists H_1-*list*, H_2-*list*, H_3-*list*, H_4-*list*, H_5-*list* and H_6-*list*, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

2. *Phase 1:* $\mathcal{B}$ responds to the queries made by $\mathcal{A}_1$ in the following ways:

   - H_1-*query*(ID_i): $\mathcal{B}$ picks $u_i \in \mathbb{Z}_q^*$ randomly, stores a new item $[ID_i, u_i]$ into H_1-*list*, and returns $g_1^{u_i}$ as the answer.
   - H_2-*query*(ID_i): $\mathcal{B}$ picks $v_i \in \mathbb{Z}_q^*$ randomly, stores a new item $[ID_i, v_i]$ into H_2-*list*, and returns $g_2^{v_i}$ as the answer. Note that if ID_i is the I-*th* distinct H_2-*query* made by $\mathcal{A}_1$, $\mathcal{B}$ returns g_2^b as the answer.
   - H_3-*query*(σ, C_1, C_2, C_3): $\mathcal{B}$ picks $\theta \in \{0,1\}^{n+l}$ randomly, stores a new item $[\sigma, C_1, C_2, C_3, \theta]$ into H_3-*list*, and returns θ as the answer.
   - H_4-*query*(M): $\mathcal{B}$ picks $h_4 \in \mathbb{G}_2$ randomly, stores a new item $[M, h_4]$ into H_4-*list*, and returns h_4 as the answer.
   - H_5-*query*(M, k): $\mathcal{B}$ picks $R \in \mathbb{Z}_q^*$ randomly, stores a new item $[M, k, R]$ into H_5-*list*, and returns R as the answer.
   - H_6-*query*(η): $\mathcal{B}$ picks $h_6 \in \mathbb{G}_2$ randomly, stores a new item $[\eta, h_6]$ into H_6-*list*, and returns h_6 as the answer.
   - *Partial private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ retrieves u_i and v_i from H_1-*list* and H_2-*list* by making H_1-*query*(ID_i) and H_2-*query*(ID_i) respectively, computes $D_{i,1} = \hat{g_1}^{u_i}$ and $D_{i,2} = \hat{g_2}^{v_i}$, and then returns $D_i = (D_{i,1}, D_{i,2})$ as the answer; otherwise, $\mathcal{B}$ aborts with failure.
   - *Private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $D_i = (D_{i,1}, D_{i,2})$ by running the algorithm *Partical private key extract* (PP, s, ID_i), computes $S_i = (S_{i,1}, S_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and sends S_i to $\mathcal{A}_1$; otherwise, $\mathcal{B}$ aborts with failure.
   - *Public key query*(ID_i): $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $X_i = \hat{g_2}^{x_i}$ and $Y_i = \hat{g_1}^{x_i}$, and then sends $PK_i = (X_i, Y_i)$ to $\mathcal{A}_1$ as the answer.
   - *Replace public key*(ID_i,PK_i'): $\mathcal{B}$ replaces the current public key with respect to ID_i with PK_i'.
   - *Decryption query*(ID_i, C): Let $C = (C_1, C_2, C_3, C_4, C_5)$.
     (a) If $i \neq I$, $\mathcal{B}$ returns the output of the algorithm *Decryption*(PP, C, SK_i) to $\mathcal{A}_1$ as the answer, where SK_i is the private key with respect to the identity ID_i.
     (b) Else, for each item $[\sigma, C_1, C_2, C_3, \theta]$ in H_3-*list*, $\mathcal{B}$ performs as follows.
       i. Compute $M' \parallel k' = C_4 \oplus \theta$.
       ii. Compute $R' = H_5(M', k')$.
       iii. Retrieve u_I from H_1-*list* by making H_1-*query*(ID_I), compute the corresponding secret value x_I by running *Set secret value*(PP), compute $S_{I,1} = g_1^{au_I x_I}$, if $C_1 = g_1^{R'}$ and $\frac{C_5}{H_4(M')^{R'}} = H_6(e(S_{I,1}, C_3))$ both hold, return M' to $\mathcal{A}_1$ as the answer.

     If there is no such item, return $\perp$ to $\mathcal{A}_1$.
   - *Authorization query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), retrieves u_i from H_1-*list* by making H_1-*query*(ID_i), computes $S_{i,1} = g_1^{au_i x_i}$, and then it returns $S_{i,1}$ as the answer; otherwise, $\mathcal{B}$ aborts with failure.

3. *Challenge:* Upon receiving two equal-length messages M_0^*,$M_1^* \in \{0,1\}^n$ and the target identity ID^*, $\mathcal{B}$ performs as follows:

   (a) If $ID^* \neq ID_I$, abort with failure.

(b) Else, pick a random bit $\rho \in \{0,1\}$ and two random numbers $k \in \{0,1\}^l$, $\beta \in \mathbb{Z}_q^*$, and then compute $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as follows:

$$\begin{cases} C_1^* = g_1^R, \\ C_2^* = g_1^c, \\ C_3^* = g_2^\beta, \\ C_4^* = (M_\rho^* \parallel k) \oplus H_3(h^{x^*}, C_1, C_2, C_3), \\ C_5^* = H_4(M_\rho^*)^R \cdot H_6(e(Q_1^*, X^{*\beta})), \end{cases}$$

where $R = H_5(M_\rho^*, k)$, $Q_1^* = H_1(ID^*)$, x^* is the corresponding secret value computed by running *Set secret value*(PP) and $PK^* = (X^*, Y^*)$ is current public key with respect to ID^*.

Finally, it sends C^* to $\mathcal{A}_1$ as the challenge ciphertext.

4. *Phase 2:* $\mathcal{A}_1$ issues as done in *Phase* 1. The constraints are that

   - ID^* does not appear in the *Private key query*;
   - If ID^*'s public key has been replaced, ID^* does not appear in the *Partial private key query*;
   - If a user's public key has been replaced, the corresponding identity ID does not appear in the *Private key query*;
   - (ID^*, C^*) does not appear in the *Decryption query*;
   - ID^* does not appear in the *Authorization query*.

5. *Guess:* $\mathcal{A}_1$ outputs a bit $\rho' \in \{0,1\}$. If $\rho = \rho'$, $\mathcal{B}$ outputs 1 for the challenge instance of the DBDH problem or 0 otherwise. □

Theorem 2 *Our proposed CL-PKEET scheme is IND-CCA secure for the authorization against Type-2 adversary (c.f. Definition 2) based on DBDH assumption in the random oracle model.*

Proof: Assume that $\mathcal{A}_2$ (a Type-2 adversary) can break our proposal, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ to solve the DBDH problem. Let $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, h)$ be an instance of the DBDH problem, $\mathcal{B}$'s task is to decide whether or not $e(g_1, g_2)^{abc} = h$ holds. $\mathcal{B}$ and $\mathcal{A}_2$ play the following game.

1. *Setup:* $\mathcal{B}$ generates the public parameter $PP = (\mathcal{G}, g_1, g_2, \hat{g_1}, \hat{g_2}, H_1, H_2, H_3, H_4, H_5, H_6)$ and the master key s, where $s \in \mathbb{Z}_q^*$ is picked randomly and $\hat{g_1} = g_1^s$, $\hat{g_2} = g_2^s$. Then $\mathcal{B}$ sends PP and s to $\mathcal{A}_2$, and picks an index $I(1 \leq I \leq q_{H_2})$ randomly as the target identity index, where $q_{H_2} \geq 0$ is the numbers of distinct H_2-*query* made by $\mathcal{A}_2$. Lists H_1-*list*, H_2-*list*, H_3-*list*, H_4-*list*, H_5-*list* and H_6-*list*, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

2. *Phase 1:* $\mathcal{B}$ responds to the queries made by $\mathcal{A}_2$ in the following ways:

   - *H_1-query, H_2-query, H_3-query, H_4-query, H_5-query and H_6-query:* $\mathcal{B}$ performs as done in the proof of *Theorem* 1.
   - *Public key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $X_i = \hat{g_2}^{x_i}$ and $Y_i = \hat{g_1}^{x_i}$, and then sends $PK_i = (X_i, Y_i)$ to $\mathcal{A}_2$ as the answer. Otherwise, $\mathcal{B}$ returns $PK_I = (X_I, Y_I)$ as the answer, where $X_I = g_2^{as}$ and $Y_I = g_1^{as}$.
   - *Private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $D_i = (D_{i,1}, D_{i,2})$ by running the algorithm *Partical private key extract* (PP, s, ID_i), computes $S_i = (S_{i,1}, S_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and sends S_i to $\mathcal{A}_2$; otherwise, $\mathcal{B}$ aborts with failure.
   - *Decryption query*(ID_i, C): Let $C = (C_1, C_2, C_3, C_4, C_5)$.
     (a) If $i \neq I$, $\mathcal{B}$ returns the output of the algorithm *Decryption*(PP, C, SK_i) to $\mathcal{A}_1$ as the answer, where SK_i is the private key with respect to the identity ID_i.
     (b) Else, for each item $[\sigma, C_1, C_2, C_3, \theta]$ in H_3-*list*, $\mathcal{B}$ performs as follows.

i. Compute $M' \parallel k' = C_4 \oplus \theta$.

ii. Compute $R' = H_5(M', k')$.

iii. Retrieve u_I from H_1-*list* by making H_1-*query*(ID_I), compute $S_{I,1} = g_1^{su_I a}$, if $C_1 = g_1^{R'}$ and $\frac{C_5}{H_4(M')^{R'}} = H_6(e(S_{I,1}, C_3))$ both hold, return M' to $\mathcal{A}_2$ as the answer.

If there is no such item, return $\perp$ to $\mathcal{A}_2$.

- *Authorization query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), retrieves u_i from H_1-*list* by making H_1-*query*(ID_i), computes $S_{i,1} = g_1^{su_i x_i}$, and then it returns $S_{i,1}$ as the answer; otherwise, $\mathcal{B}$ aborts with failure.

3. *Challenge:* Upon receiving two equal-length messages M_0^*,$M_1^* \in \{0,1\}^n$ and the target identity ID^*, $\mathcal{B}$ performs as follows:

   (a) If $ID^* \neq ID_I$, abort with failure.

   (b) Else, pick a random bit $\rho \in \{0,1\}$ and two random numbers $k \in \{0,1\}^l$, $\beta \in \mathbb{Z}_q^*$, and then compute $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ as follows:

$$\begin{cases} C_1^* = g_1^R, \\ C_2^* = g_1^c, \\ C_3^* = g_2^\beta, \\ C_4^* = (M_\rho^* \parallel k) \oplus H_3(h^s, C_1, C_2, C_3), \\ C_5^* = H_4(M_\rho^*)^R \cdot H_6(e(Q_1^*, X^{*\beta})), \end{cases}$$

   where $R = H_5(M_\rho^*, k)$, $Q_1^* = H_1(ID^*)$, $PK^* = (X^*, Y^*)$ is current public key with respect to ID^*.

   Finally, it sends C^* to $\mathcal{A}_2$ as the challenge ciphertext.

4. *Phase 2:* $\mathcal{A}_2$ issues as done in *Phase* 1. The constraints are that

   - ID^* does not appear in the *Private key query*;
   - (ID^*, C^*) does not appear in the *Decryption query*;
   - ID^* does not appear in the *Authorization query*.

5. *Guess:* $\mathcal{A}_2$ outputs a bit $\rho' \in \{0,1\}$. If $\rho = \rho'$, $\mathcal{B}$ outputs 1 for the challenge instance of the DBDH problem or 0 otherwise. □

Theorem 3 *Our proposed CL-PKEET scheme is OW-CCA secure for the authorization against Type-3 adversary (c.f. Definition 3) based on DBDH assumption in the random oracle model.*

Proof: Assume that $\mathcal{A}_3$ (a Type-3 adversary) can break our proposal, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ to solve the DBDH problem. Let $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, h)$ be an instance of the DBDH problem, $\mathcal{B}$'s task is to decide whether or not $e(g_1, g_2)^{abc} = h$ holds. $\mathcal{B}$ and $\mathcal{A}_3$ play the following game.

1. *Setup:* $\mathcal{B}$ generates the public parameter $PP = (\mathcal{G}, g_1, g_2, \hat{g_1}, \hat{g_2}, H_1, H_2, H_3, H_4, H_5, H_6)$, where $\hat{g_1} = g_1^a$ and $\hat{g_2} = g_2^a$. Then $\mathcal{B}$ sends PP to $\mathcal{A}_3$, and picks an index $I (1 \leq I \leq q_{H_2})$ randomly as the target identity index, where $q_{H_2} \geq 0$ is the numbers of distinct H_2-*query* made by $\mathcal{A}_3$. Lists H_1-*list*, H_2-*list*, H_3-*list*, H_4-*list*, H_5-*list* and H_6-*list*, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

2. *Phase 1:* $\mathcal{B}$ responds to the queries made by $\mathcal{A}_3$ in the following ways:

   - *H_1-query, H_2-query, H_3-query, H_4-query, H_5-query and H_6-query:* $\mathcal{B}$ performs as done in the proof of *Theorem* 1.
   - *Partial private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ retrieves u_i and v_i from H_1-*list* and H_2-*list* by making H_1-*query*(ID_i) and H_2-*query*(ID_i) respectively, computes $D_{i,1} = \hat{g_1}^{u_i}$ and $D_{i,2} = \hat{g_2}^{v_i}$, and then returns $D_i = (D_{i,1}, D_{i,2})$ as the answer; otherwise, $\mathcal{B}$ aborts with failure.

- *Private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $D_i = (D_{i,1}, D_{i,2})$ by running the algorithm *Partical private key extract* (PP, s, ID_i), computes $S_i = (S_{i,1}, S_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and sends S_i to $\mathcal{A}_3$; otherwise, $\mathcal{B}$ aborts with failure.
- *Public key query*(ID_i): $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $X_i = \hat{g_2}^{x_i}$ and $Y_i = \hat{g_1}^{x_i}$, and then sends $PK_i = (X_i, Y_i)$ to $\mathcal{A}_3$ as the answer.
- *Replace public key*(ID_i,PK_i'): $\mathcal{B}$ replaces the current public key with respect to ID_i with PK_i'.
- *Decryption query*(ID_i, C): Let $C = (C_1, C_2, C_3, C_4, C_5)$.
  (a) If $i \neq I$, $\mathcal{B}$ returns the output of the algorithm *Decryption*(PP, C, SK_i) to $\mathcal{A}_3$ as the answer, where SK_i is the private key with respect to the identity ID_i.
  (b) Else, for each item $[\sigma, C_1, C_2, C_3, \theta]$ in H_3-*list*, $\mathcal{B}$ performs as follows.
     i. Compute $M' \parallel k' = C_4 \oplus \theta$.
     ii. Compute $R' = H_5(M', k')$.
     iii. Compute the corresponding trapdoor $S_{I,1}$ by running *Authorization*(ID_I), if $C_1 = g_1^{R'}$ and $\frac{C_5}{H_4(M')^{R'}} = H_6(e(S_{I,1}, C_3))$ both hold, return M' to $\mathcal{A}_3$ as the answer.

     If there is no such item, return $\perp$ to $\mathcal{A}_3$.
- *Authorization query*(ID_i): $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), retrieves u_i from H_1-*list* by making H_1-*query*(ID_i), computes $S_{i,1} = g_1^{au_ix_i}$, and then it returns $S_{i,1}$ as the answer.

3. *Challenge:* $\mathcal{B}$ picks a message M^* and two numbers $k \in \{0,1\}^l$, $\beta \in \mathbb{Z}_q^*$ randomly, and then computes $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ with the target identity ID_I as follows:
$$\begin{cases} C_1^* = g_1^R, \\ C_2^* = g_1^c, \\ C_3^* = g_2^\beta, \\ C_4^* = (M^* \parallel k) \oplus H_3(h^{x_I}, C_1, C_2, C_3), \\ C_5^* = H_4(M^*)^R \cdot H_6(e(Q_1^*, {X_I}^\beta)), \end{cases}$$
where $R = H_5(M^*, k)$, $Q_1^* = H_1(ID_I)$, x_I is the corresponding secret value computed by running *Set secret value*(PP) and $PK_I = (X_I, Y_I)$ is current public key with respect to ID_I.

   Finally, it sends C^* to $\mathcal{A}_3$ as the challenge ciphertext.

4. *Phase 2:* $\mathcal{A}_3$ issues as done in *Phase* 1. The constraints are that
   - ID^* does not appear in the *Private key query*;
   - If ID^*'s public key has been replaced, ID^* does not appear in the *Partial private key query*;
   - If a user's public key has been replaced, the corresponding identity ID does not appear in the *Private key query*;
   - (ID^*, C^*) does not appear in the *Decryption query*.

5. *Guess:* $\mathcal{A}_3$ outputs $M^{'}$. If $M^* = M^{'}$, $\mathcal{B}$ outputs 1 for the challenge instance of the DBDH problem or 0 otherwise. □

Theorem 4 *Our proposed CL-PKEET scheme is OW-CCA secure for the authorization against Type-4 adversary (c.f. Definition 4) based on DBDH assumption in the random oracle model.*

Proof: Assume that $\mathcal{A}_4$ (a Type-4 adversary) can break our proposal, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ to solve the DBDH problem. Let $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b, h)$ be an instance of the DBDH problem, $\mathcal{B}$'s task is to decide whether or not $e(g_1, g_2)^{abc} = h$ holds. $\mathcal{B}$ and $\mathcal{A}_4$ play the following game.

1. *Setup:* $\mathcal{B}$ generates the public parameter $PP = (\mathcal{G}, g_1, g_2, \hat{g_1}, \hat{g_2}, H_1, H_2, H_3, H_4, H_5, H_6)$ and the master key s, where $s \in \mathbb{Z}_q^*$ is picked randomly and $\hat{g_1} = g_1^s$, $\hat{g_2} = g_2^s$. Then $\mathcal{B}$ sends PP and s to $\mathcal{A}_4$, and picks an index $I(1 \leq I \leq q_{H_2})$ randomly as the target identity index, where $q_{H_2} \geq 0$ is the numbers of distinct H_2-*query* made by $\mathcal{A}_4$. Lists H_1-*list*, H_2-*list*, H_3-*list*, H_4-*list*, H_5-*list* and H_6-*list*, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

2. *Phase 1:* $\mathcal{B}$ responds to the queries made by $\mathcal{A}_4$ in the following ways:

   - *H_1-query, H_2-query, H_3-query, H_4-query, H_5-query and H_6-query:* $\mathcal{B}$ performs as done in the proof of *Theorem* 1.
   - *Public key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $X_i = \hat{g_2}^{x_i}$ and $Y_i = \hat{g_1}^{x_i}$, and then sends $PK_i = (X_i, Y_i)$ to $\mathcal{A}_4$ as the answer; otherwise, $\mathcal{B}$ returns $PK_I = (X_I, Y_I)$ as the answer, where $X_I = g_2^{as}$ and $Y_I = g_1^{as}$.
   - *Private key query*(ID_i): If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), computes $D_i = (D_{i,1}, D_{i,2})$ by running the algorithm *Partical private key extract* (PP, s, ID_i), computes $S_i = (S_{i,1}, S_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and sends S_i to $\mathcal{A}_4$; otherwise, $\mathcal{B}$ aborts with failure.
   - *Decryption query*(ID_i, C): Let $C = (C_1, C_2, C_3, C_4, C_5)$.
     (a) If $i \neq I$, $\mathcal{B}$ returns the output of the algorithm *Decryption*(PP, C, SK_i) to $\mathcal{A}_4$ as the answer, where SK_i is the private key with respect to the identity ID_i.
     (b) Else, for each item $[\sigma, C_1, C_2, C_3, \theta]$ in H_3-*list*, $\mathcal{B}$ performs as follows.
       i. Compute $M' \parallel k' = C_4 \oplus \theta$.
       ii. Compute $R' = H_5(M', k')$.
       iii. Compute the corresponding trapdoor $S_{I,1}$ by running *Authorization*(ID_I), if $C_1 = g_1^{R'}$ and $\frac{C_5}{H_4(M')^{R'}} = H_6(e(S_{I,1}, C_3))$ both hold, return M' to $\mathcal{A}_4$ as the answer.

     If there is no such item, return $\perp$ to $\mathcal{A}_4$.
   - *Authorization query*(ID_i): $\mathcal{B}$ retrieves u_i from H_1-*list* by making H_1-*query*(ID_i). If $i \neq I$, $\mathcal{B}$ computes the corresponding secret value x_i by running *Set secret value*(PP), and computes $S_{i,1} = g_1^{su_ix_i}$; otherwise, $\mathcal{B}$ computes $S_{i,1} = g_1^{au_is}$. Then, it returns $S_{i,1}$ to $\mathcal{A}_4$ as the answer.

3. *Challenge:* $\mathcal{B}$ picks a message M^* and two numbers $k \in \{0,1\}^l$, $\beta \in \mathbb{Z}_q{}^*$ randomly, and then computes $C^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*)$ with the target identity ID_I as follows:
$$\begin{cases} C_1^* = g_1^R, \\ C_2^* = g_1^c, \\ C_3^* = g_2^\beta, \\ C_4^* = (M^* \parallel k) \oplus H_3(h^s, C_1, C_2, C_3), \\ C_5^* = H_4(M^*)^R \cdot H_6(e(Q_1^*, X_I{}^\beta)), \end{cases}$$
   where $R = H_5(M^*, k)$, $Q_1^* = H_1(ID_I)$, $PK_I = (X_I, Y_I)$ is current public key with respect to ID_I.
   Finally, it sends C^* to $\mathcal{A}_4$ as the challenge ciphertext.

4. *Phase 2:* $\mathcal{A}_4$ issues as done in *Phase* 1. The constraints are that

   - ID^* does not appear in the *Private key query*;
   - (ID^*, C^*) does not appear in the *Decryption query*;

5. *Guess:* $\mathcal{A}_4$ outputs $M^{'}$. If $M^* = M^{'}$, $\mathcal{B}$ outputs 1 for the challenge instance of the DBDH problem or 0 otherwise. □

6 Conclusion

In this paper, we propose the concept of certificateless public key encryption with equality test (CL-PKEET) to solve the key escrow problem in IBEET. More in details, we first give the definition of CL-PKEET and define the security model. Finally, we propose a concrete CL-PKEET scheme based on the DBDH assumption and prove its security.

References

[1] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. *Lecture Notes in Computer Science*, 2894(2):452–473, 2003.

[2] J. Baek, R. Safavinaini, and W. Susilo. Public key encryption with keyword search revisited. In *International Conference on Computational Science and Its Applications*, 2008.

[3] S. Chatterjee and A. Menezes. On cryptographic protocols employing asymmetric pairings-the role of revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

[4] B. Dan, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. *Public Key Encryption with Keyword Search*. Springer Berlin Heidelberg, 2004.

[5] S. T. Hsu, C. C. Yang, and M. S. Hwang. A study of public key encryption with keyword search. *International Journal of Network Security*, 15(2):71–79, 2013.

[6] K. Huang, R. Tso, Y. C. Chen, S. M. M. Rahman, A. Almogren, and A. Alamri. Pke-aet: Public key encryption with authorized equality test. *The Computer Journal*, 58(10), 2015.

[7] K. Lee, J. H. Park, and H. L. Dong. Anonymous hibe with short ciphertexts: full security in prime order groups. *Designs, Codes and Cryptography*, 74(2):395–425, 2015.

[8] S. Ma. Public key encryption with delegated equality test in a multi-user setting. *Computer Journal*, 58(4), 2015.

[9] S. Ma. Identity-based encryption with outsourced equality test in cloud computing. *Information Sciences An International Journal*, 328(C):389–402, 2016.

[10] S. Ma, Q. Huang, M. Zhang, and B. Yang. Efficient public key encryption with equality test supporting flexible authorization. *IEEE Transactions on Information Forensics & Security*, 10(3):458–470, 2015.

[11] A. Shamir. Identity-based cryptosystems and signature schemes. *Lecture Notes in Computer Science*, 21(2):47–53, 1995.

[12] Q. Tang. Public key encryption supporting plaintext equality test and user-specified authorization. *Security & Communication Networks*, 5(12):1351–1362, 2011.

[13] Q. Tang. Towards public key encryption scheme supporting equality test with fine-grained authorization. In *Australasian Conference on Information Security and Privacy*, pages 389–406, 2011.

[14] Q. Tang. Public key encryption schemes supporting equality test with authorisation of different granularity. *International Journal of Applied Cryptography*, 2(4):304–321, 2012.

[15] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong. Probabilistic public key encryption with equality test. In *International Conference on Topics in Cryptology*, pages 119–131, 2010.