ITC 4/52

Information Technology and Control
Vol. 52 / No. 4 / 2023
pp. 935-951
DOI 10.5755/j01.itc.52.4.33765

CBEET: Constructing Certificate-based Encryption with Equality Test in the CB-PKS

Received 2023/04/03 | Accepted after revision 2023/10/19

HOW TO CITE: Tsai, T.-T., Lin, H.-Y., Wu, C.-Y. (2023). CBEET: Constructing Certificate-based Encryption with Equality Test in the CB-PKS. *Information Technology and Control*, 52(4), 935-951. https://doi.org/10.5755/j01.itc.52.4.33765

CBEET: Constructing Certificate-based Encryption with Equality Test in the CB-PKS

Tung-Tso Tsai, Han-Yu Lin, Cheng-Ye Wu
Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung 202, Taiwan

Corresponding author: hanyu@email.ntou.edu.tw

To maintain the confidentiality of private data, encryption mechanisms have become prevalent. Researchers always strive to design secure and efficient encryption mechanisms in both symmetric and asymmetric key systems. Certificate-based public key systems (CB-PKS) belong to the family of asymmetric key systems. CB-PKS offers solutions to both the key escrow problem present in identity-based public key systems, and the need to construct a public key infrastructure in traditional public key systems. The past saw a wealth of research into the encryption mechanisms in the CB-PKS, called certificate-based encryption (CBE). Indeed, encrypted data (ciphertext) can be used in other applications such as the comparison of personal medical data as two ciphertexts can be compared to determine if they contain the same data (plaintext). However, the equality test of two ciphertexts in the CB-PKS is an open issue since research which has empirically studied is scant. The purpose of this paper is to propose the first certificate-based encryption with equality test (CBEET), and to prove that it is secure under the bilinear Diffie-Hellman (BDH) assumption.

KEYWORDS: Certificate-based cryptography, encryption, equality test, bilinear pairing.

1. Introduction

In traditional public key systems [1, 3, 31], a sender can use a receiver's public key to encrypt a message (data) and send it to the receiver via public channels. However, the relationship between the public key and the receiver's identity must be confirmed before such encryption procedure is executed, because the public key is composed of arbitrary random numbers. The purpose of establishing public key infrastructures (PKI) is to connect the public key with the receiver's identity. To avoid the construction of PKI, an identity-based public key system (ID-PKS) was proposed by Shamir [18], in which the receiver's public key can be regarded as her/his identity such as social security number or e-mail address. Based on the concept of ID-PKS, many identity-based encryption (IBE) schemes [2, 4, 17, 22, 23] have been proposed.

A concept of public key encryption with keyword search (PKEKS) was proposed by Boneh et al. [3], in which users can upload an encrypted keyword (ciphertext) to the cloud server, and the ciphertext can be found by the cloud server with the authorizations from users. It is worth noting that the cloud server cannot know the message (plaintext) in the ciphertext. However, PKEKS has a restriction that the cloud server can only target a single user in the search process in the sense that all the ciphertexts that can be successfully searched must be encrypted from the same public key. To overcome this restriction, Yang et al. [28] proposed public key encryption with equality test (PKEET), in which the cloud server can use the comparison of ciphertexts to search the target ciphertext, while ciphertexts can be encrypted under different public keys.

Based on the first PKEET [28], Tang [21] proposed a new PKEET, in which the property of fine-gained authorization was considered to enable users to own this authorization over ciphertexts. To clearly point out the specific users who can perform the equality test for ciphertexts, Ma et al. [16] presented a delegation mechanism for PKEET, namely public key encryption with delegated equality test (PKE-DET). To filter testable ciphertexts, public key encryption with filtered equality test (PKE-FET) was proposed by Huang et al. [11]. For enhancing the security of PKEET, Duong et al. [5] proposed a PKEET in the standard model, while the PKEET is secure under the hardness assumption of lattices. Zeng et al. [32] hired the hash proof system to propose a secure PKEET in the standard model. Recently, the work of PKEET in the standard model are still valued and several PKEETs have been proposed in the literature [6, 12].

Indeed, PKEET is a cryptographic mechanism designed under the traditional public key systems (PKS) which rely on the construction of PKI to manage certificates. To tackle the problem, Ma [15] proposed a new scheme, namely identity-based with equality test (IBEET) in the ID-PKS. Although Ma [15] presented the security analysis for the IBEET scheme, Liao et al. [14] pointed out that the scheme is insecure. For improving the performance on computational cost, Wu et al. [26] proposed an efficient IBEET scheme. In the above IBEET schemes, there is only one type of authorization from users to the cloud server. To make the way of authorization more flexible, three types of authorization were presented and used into IBEET [13]. In addition, to resist the attack of quantum computing, an IBEET scheme from lattices was proposed by Zhang and Xu [33].

However, the ID-PKS had a drawback, namely, the key escrow problem since each user's private key is generated by the key generator center (KGC) of the ID-PKS. To remove the key escrow problem, certificate-based public key system (CB-PKS) was proposed by Gentry [8], which not only eliminates the key escrow problem in the ID-PKS, but also avoids the construction of PKI in traditional public key systems. Based on the CB-PKS, a large number of certificate-based encryption (CBE) schemes have been published in the literature [7, 9, 19, 25, 29, 30] to protect the confidentiality of private data. Research related to CBE continues to be explored to this day. Shareef and Sagheer [20] presented an enhanced CBE scheme for big data environments by combining AES and (ECDSA-ECDH) techniques. The goal of this scheme is to improve encryption efficiency and security in big data scenarios. In order to avoid relying on random oracles for security analysis, Guo et al. [10] introduced an efficient CBE scheme without random oracles. On the other hand, to enable secure keyword searching in cloud environments, Uyyala [24] presented a secure channel-independent certificate-based searchable encryption scheme that can withstand outside and inside keyword guessing attacks. To protect data security under leakage risks while satisfying the efficiency and scalability requirements of cloud computing environments, Zhou et al. [34] introduced a continuous leakage-resilient certificate-based signcryption scheme applied in cloud computing.

As mentioned in the previous PKEETs and IBEETs, the encrypted data (ciphertext) can be used in other applications such as the comparison of personal medical data since two ciphertexts can be compared whether they contain the same plaintext. Relevant PKEET scheme [31] and IBEET scheme [13] have been proposed in both traditional PKS and ID-PKS, respectively. To the best of our knowledge, there is no related work on equality test of ciphertexts in the CB-PKS. Table 1 lists the comparisons between the PKE scheme [31], the PKEET scheme [6], IBE scheme [23], IBEET scheme [13], CBE scheme [25] and our CBEET scheme in terms of public key setting,

Table 1
Comparisons between the existing schemes and our CBEET scheme

Schemes	Public-key setting	Eliminating PKI construction	Avoiding key escrow problem	Possessing equality test property
Yu et al.'s PKE scheme [31]	PKI-based	No	Yes	No
Duong et al.'s PKEET scheme [6]	PKI-based	No	Yes	Yes
Tseng et al.'s IBE scheme [23]	ID-based	Yes	No	No
Li et al.'s IBEET scheme [13]	ID-based	Yes	No	Yes
Wu et al.'s CBE scheme [25]	Certificate-based	Yes	Yes	No
Our CBEET cheme	Certificate-based	Yes	Yes	Yes

eliminating PKI construction, avoiding key escrow problem and possessing equality test property. After comparing with other existing schemes, we attempt to propose the first CBEET scheme that provides the equality test of ciphertexts in the CB-PKS.

Although the existing IBEET schemes [13, 15, 26] possessed the property of equality test of ciphertexts, they cannot avoid the key escrow problem. On the other hand, a well-known fact is that the existing CBE schemes [9, 25, 30] can avoid the key escrow problem. However, these CBE schemes do not provide a mechanism for equality test. In this paper, we extend CBE schemes to propose the first propose the *first* certificate-based encryption with equality test. Several specific contributions are shown as below.

- We extend the syntax and security notions of CBE and consider the property of equality test of ciphertexts to define a novel syntax and security notions of CBEET.
- A concrete CBEET scheme is proposed under the syntax of CBEET.
- We demonstrate that the proposed CBEET scheme is secure under the bilinear Diffie-Hellman (BDH) assumption.
- Compared with the existing schemes, the proposed CBEET scheme not only retains the efficiency of encryption and decryption, but also has the mechanism of equality test.

The rest of this paper is as follows. Preliminaries are given in Section 2. Section 3 shows the definitions of the syntax and security notions. A concrete CBEET scheme is presented in Section 4. Section 5 analyzes the security of the CBEET scheme. A comparison and a conclusion are given in Sections 6 and 7, respectively.

2. Preliminaries

Two preliminaries including the concept of bilinear pairings [4] and the bilinear Diffie-Hellman (BDH) assumption [15] are introduced in this section. Two definitions related to bilinear pairings and the BDH assumption are given as below.

Definition 1. (Bilinear pairings). Assume that q is large prime number and it is the order for three multiplicative cyclic groups, namely G_1, G_2 and G_T. We say that e: $G_1 \times G_2 \rightarrow G_T$ is an asymmetric bilinear pairing if e has the following three properties.

1. Bilinearity: for any x, y Z_q^*, $e(U^x, V^y) = e(U, V)^{xy}$, where U and V respectively are generators in G_1 and G_2.
2. Non-degeneracy: $e(U, V) \neq 1$, where the definitions of U and V are the same as (1) above.
3. Computability: $e(g_1, g_2)$ is efficiently computable for all $g_1 \in G_1, g_2 \in G_2$.

Definition 2. (BDH assumption). Assume that $\mathcal{G} = (q, G_1, G_2, G_T, e, U, U^a, U^c, V, V^a, V^b)$ is the instance of the BDH problem, where a, b, $c \in Z_q^*$ are unknown for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$. Then, $\mathcal{A}$'s advantage of computing $e(U, V)^{abc}$ is negligible and defined as $\Pr[\mathcal{A}(U, U^a, U^c, V, V^a, V^b) = e(U, V)^{abc}] < \epsilon$.

To improve readability for readers, we have compiled Table 2, which contains a comprehensive list of symbols employed in the proposed CBEET.

Table 2
Symbols

Symbol	Meaning
SSK	The system secret key
SPP	The system public parameters
USK	The user secret key
UPK_{pair}	The user public key pair
$UPKA$	The first part of user public key pair
$UPKB$	The second part of user public key pair
$Cert_{pair}$	The user certificate pair
$CertA$	The first part of user certificate pair
$CertB$	The second part of user certificate pair
M	The message
CT	The ciphertext
TD	The trapdoor

3. Syntax and Security Notions

Inspired by previous works related to CBE [7, 29] and the properties of equality test [5, 15], we define a new syntax and new security notions of CBEET. Based on the syntax of CBE, we present the new syntax of CBEET by adding the Trapdoor and Test algorithms. Following the security notions of CBE, we present the new security notions of CBEET by adding two new types of adversaries who can obtain the trapdoor from the Trapdoor algorithm.

3.1. Syntax of CBEET

There are three roles: the certification authority (CA), the users (sender/receiver) and the cloud server (CS) in the syntax of CBEET. We use Figure 1 to depict the generation of user's full secret key and public key. The CA performs the *Setup* algorithm to generate the system secret key SSK and system public parameters SPP. A user can set her/his user secret key USK and user public key pair $UPK_{pair} = (UPKA, UPKB)$ according to the system public parameters SPP. Immediately, the user sends her/his identity ID and user public key pair UPK_{pair} to the CA. The user certificate pair $Cert_{pair}$ can be calculated by the CA. And, $Cert_{pair}$ is returned to the user via secure channels. Notice that the user's full secret key consists of user secret key USK and user certificate pair $Cert_{pair}$. Figure 2 describes the process of encryption and decryption. A sender can use the identity ID and public key pair UPK_{pair} of the receiver to encrypt the plaintext (message M) and

Figure 1
Generation of user's full secret key and public key

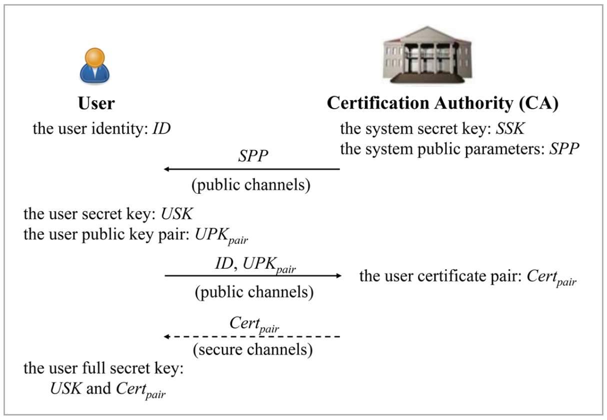

User
the user identity: ID
Certification Authority (CA)
the system secret key: SSK
the system public parameters: SPP
SPP
(public channels)
the user secret key: USK
the user public key pair: UPK_pair
ID, UPK_pair
(public channels)
the user certificate pair: Cert_pair
Cert_pair
(secure channels)
the user full secret key:
USK and Cert_pair

Figure 2
The process of encryption and decryption

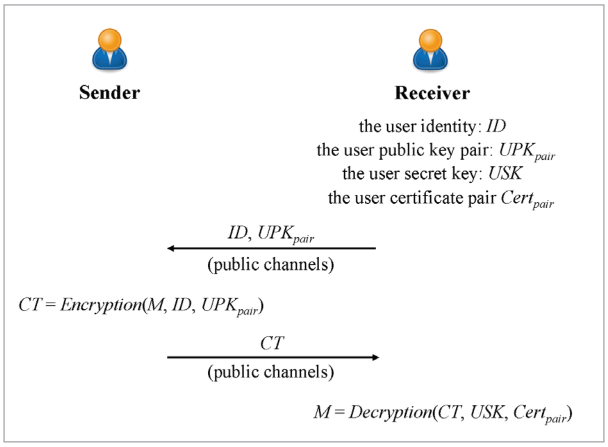
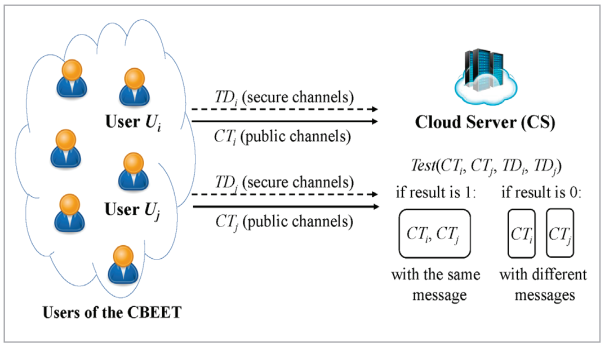

Sender
Receiver
the user identity: ID
the user public key pair: UPK_pair
the user secret key: USK
the user certificate pair Cert_pair
ID, UPK_pair
(public channels)
CT = Encryption(M, ID, UPK_pair)
CT
(public channels)
M = Decryption(CT, USK, Cert_pair)

Figure 3
The work of the CS for testing



obtain the ciphertext CT. After receiving the ciphertext CT, the receiver can use the associated USK and $Cert_{pair}$ to decrypt and obtain the plaintext. Figure 3 introduces the work of the CS for testing. Each user can send her/his own trapdoor and ciphertexts to the CS who can compare the ciphertexts to confirm whether any two ciphertexts contain the same plaintext. Next, we formally introduce the syntax of CBEET which includes the following seven algorithms.

- ***Setup***: The CA inputs a security parameter λ to gain the system secret key *SSK* and system public parameters *SPP*.
- ***UserKeyGen***: The user inputs the system public parameters *SPP* to gain her/his user secret key *USK* and user public key pair UPK_{pair} = (*UPKA*, *UPKB*). Moreover, the user sends her/his UPK_{pair} to the CA.
- ***CertGen***: The CA inputs the system public parameters *SPP*, user's identity *ID*, system secret key *SSK* and user public key pair UPK_{pair} to calculate the user's certificate pair $Cert_{pair}$. In addition, the CA sends $Cert_{pair}$ to the user via a secure channel.
- ***Encryption***: A sender inputs the system public parameters *SPP*, user's identity *ID*, user public key pair UPK_{pair} and a message $M \in \{0, 1\}^{\lambda}$ to generate a ciphertext *CT*.
- ***Decryption***: The receiver inputs the system public parameters *SPP*, the ciphertext *CT*, her/his user secret key *USK* and $Cert_{pair}$ to obtain the message *M*.
- ***Trapdoor***: The user inputs public parameters *PP*, her/his $Cert_{pair}$ and user secret key *USK* to obtain a trapdoor *TD*. Then, the user sends *TD* to the CS via a secure channel.
- ***Test***: For any two users U_i and U_j, the CS inputs the system public parameters *SPP*, two ciphertexts CT_i, CT_j and two trapdoors TD_i, TD_j to return 1 or 0.

3.2. Security Notions of CBEET

We know that there are two types of adversaries in the CB-PKC systems [7, 29]. One is a non-system member (external adversary), and the other is the CA who could be honest-but-curious since the CA possesses the system secret key *SSK*. In order to satisfy the equality test properties [5, 15], we must consider the situation that adversaries can obtain the trapdoor *TD* in the security notions. The following are four types of adversaries in the proposed CBEET.

- Type I adversary is an external adversary who has the ability of replacing the user public key pair UPK_{pair}.
- Type II adversary is the CA who has the system secret key *SSK* which is used to calculate the user's certificate pair $Cert_{pair}$.
- Type III adversary is the same as Type I adversary, except that the trapdoor *TD* can be obtained.
- Type IV adversary is the same as Type II adversary, except that the trapdoor *TD* can be obtained.

Based on the security notions of CBE [7, 29], we add the equality test properties [5, 15] to define two new security games (later presented in Definitions 3 and 4) for the CBEET. The first security game named the $G_{\text{CBEET-IND-CCA}}$ is used to model indistinguishabilty under chosen ciphertext attacks from Type I and Type II adversaries. The second security game named the $G_{\text{CBEET-OW-CCA}}$ is used to model one-wayness under chosen ciphertext attacks from Type III and Type IV adversaries.

Definition 3 **($G_{\text{CBEET-IND-CCA}}$).** Assume that $\mathcal{A}$ is the PPT adversary (including Type I and Type II) for a CBEET scheme. We say that the scheme is secure for indistinguishabilty under chosen ciphertext attacks if $\mathcal{A}$'s advantage of winning the following game with a challenger $\mathcal{C}$ can be negligible.

- **Setup.** The challenger $\mathcal{C}$ performs the Setup of CBEET with a security parameter λ to gain the system secret key *SSK* and system public parameters *SPP*. The challenger $\mathcal{C}$ sends *SPP* to the adversary $\mathcal{A}$. Further, the *SSK* will also be transmitted if $\mathcal{A}$ is Type II adversary.
- **Phase 1.** The adversary $\mathcal{A}$ may issue the following queries.
  - *User secret key query*: When receiving this query with an identity *ID*, the challenger $\mathcal{C}$ performs the UserKeyGen of CBEET to gain user secret key *USK* and user public key pair UPK_{pair}. Then, $\mathcal{C}$ returns the *USK* to $\mathcal{A}$ if the identity *ID* did not appear in *User public key replace query*.
  - *User public key query*: When receiving this query with an identity *ID*, the challenger $\mathcal{C}$ returns the associated user public key pair UPK_{pair}.
  - *User public key replace query*: When receiving this query with an identity *ID* and UPK'_{pair}, the challenger $\mathcal{C}$ replaces the user public key pair UPK_{pair} related to the identity *ID* with UPK'_{pair}.
  - *Certification query*: When receiving this query with an identity *ID* and the associated user public key pair UPK_{pair}, the challenger $\mathcal{C}$ performs the

CertGen of CBEET to gain the certificate pair $Cert_{pair}$. Then, $\mathcal{C}$ returns the $Cert_{pair}$ to $\mathcal{A}$.

- *Decryption query*: When receiving this query with an identity ID and the associated ciphertext CT, the challenger $\mathcal{C}$ performs the Decryption of CBEET to gain the resulting message M. Then, $\mathcal{C}$ returns it to $\mathcal{A}$.
- *Trapdoor query*: When receiving this query with an identity ID, the challenger $\mathcal{C}$ performs the Trapdoor of CBEET to gain the resulting trapdoor TD. Then, $\mathcal{C}$ returns it to $\mathcal{A}$.
- **Challenge.** The adversary $\mathcal{A}$ sends a message pair (M_0^*, M_1^*) and a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ randomly chooses a random value $rv \in \{0, 1\}$ to perform the Encryption of CBEET with SPP, ID^*, UPK^*_{pair} and M_{rv}^* to obtain a challenge ciphertext CT^*. Then, $\mathcal{C}$ returns CT^* to $\mathcal{A}$. In addition, three restrictions must be satisfied as below.
- Whether the adversary $\mathcal{A}$ belongs to Type I or Type II, ID^* cannot appear in the *Trapdoor query*.
- If the adversary $\mathcal{A}$ belongs to Type I, ID^* cannot appear in the *Certification query*.
- If the adversary $\mathcal{A}$ belongs to Type II, ID^* cannot appear in both the *User secret key query* and *User public key replace query*.

– **Phase 2.** The adversary $\mathcal{A}$ can continue to issue the same queries as in phase 1.

– **Guess.** The adversary $\mathcal{A}$ sends a value $rv' \in \{0, 1\}$ and wins this game $G_{\text{CBEET-IND-CCA}}$ if $rv' = rv$. Here, we denote $\mathcal{A}$'s advantage as $\text{Adv}_{\mathcal{A}}(\lambda) = |\Pr[rv = rv'] - 1/2|$.

Definition 4 **($G_{\text{CBEET-OW-CCA}}$).** Assume that $\mathcal{A}$ is the PPT adversary (including Type III and Type IV) for a CBEET scheme. We say that the scheme is secure for one-wayness under chosen ciphertext attacks if $\mathcal{A}$'s advantage of winning the following game with a challenger $\mathcal{C}$ can be negligible.

– **Setup.** The challenger $\mathcal{C}$ performs the Setup of CBEET with a security parameter λ to gain the system secret key SSK and system public parameters SPP. The challenger $\mathcal{C}$ sends SPP to the adversary $\mathcal{A}$. Further, the SSK will also be transmitted if $\mathcal{A}$ is Type IV adversary.

– **Phase 1.** This phase is the same as in the phase 1 of the game $G_{\text{CBEET-IND-CCA}}$.

– **Challenge.** The adversary $\mathcal{A}$ sends a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ randomly chooses a random message M^* to perform the Encryption of CBEET with SPP, ID^*, UPK^*_{pair} and M^* to obtain a challenge ciphertext CT^*. Then, $\mathcal{C}$ returns CT^* to $\mathcal{A}$. In addition, two restrictions must be satisfied as below.

- If the adversary $\mathcal{A}$ belongs to Type III, ID^* cannot appear in the *Certification query*.
- If the adversary $\mathcal{A}$ belongs to Type IV, ID^* cannot appear in both the *User secret key query* and *User public key replace query*.

– **Phase 2.** The adversary $\mathcal{A}$ can continue to issue the same queries as in phase 1.

Guess. The adversary $\mathcal{A}$ sends a message M' and wins this game $G_{\text{CBEET-OW-CCA}}$ if $M' = M^*$. Here, we denote $\mathcal{A}$'s advantage as $\text{Adv}_{\mathcal{A}}(\lambda) = |\Pr[M^* = M'] - 1/2|$.

4. Concrete CBEET Scheme

Our proposed CBEET scheme, as shown in Figure 4, consists of three roles: the CA, the users (sender/receiver), and the CS. The CA executes both the Setup and CertGen algorithms, while the users are responsible for the UserKeyGen, Encryption, Decryption, and Trapdoor algorithms. The CS takes charge of executing the Test algorithm. We can obtain the detailed process of executing these algorithms from the following.

– **Setup:** The CA inputs a security parameter λ to gain $\mathcal{G} = (q, G_1, G_2, G_T, e)$ defined in Section 2. Then, the CA sets the system secret key $SSK = s$ and system public parameters $SPP = (\mathcal{G}, U, V, SPK, H_1, H_2, H_3, H_4, H_5, H_6)$ by performing the following steps.

- Randomly choose a value $s \in Z_q^*$, and set $SSK = s$.
- Pick two generators $U \in G_1$ and $V \in G_2$, and calculate system public key $SPK = U^s$.
- Set six hash functions: H_1: $\{0, 1\}^* \times G^2{}_1 \rightarrow G_2$, H_2: $\{0, 1\}^* \times G^2{}_1 \rightarrow G_2$, H_3: $G_T \times G^2{}_1 \rightarrow \{0, 1\}^{\lambda+l}$, H_4: $\{0, 1\}^{\lambda} \rightarrow G_2$, H_5: $\{0, 1\}^{\lambda+l} \rightarrow Z_q^*$, H_6: $G_T \rightarrow G_2$, where l is a fixed length.

Note that the CA will securely store the SSK to maintain its confidentiality, while the SPP will be made publicly available to all users.

– **UserKeyGen**: The user inputs the system public parameters SPP to gain V and SPK. Then, the user randomly chooses a value $x \in Z_q^*$ to set her/his user secret key $USK = x$ and user public key pair $UPK_{pair} = (UPKA, UPKB) = (SPK^x, V^x) = (U^{sx}, V^x)$. Moreover, the user sends her/his UPK_{pair} to the CA. Here, the user will securely store the user secret key USK to maintain its confidentiality, while the user public key pair UPK_{pair} will be made publicly available to all users.

- **CertGen**: The CA inputs the system public parameters SPP, an identity ID, system secret key SSK and user public key pair UPK_{pair} to calculate the user's certificate pair $Cert_{pair} = (CertA, CertB)$, where $CertA = H_1(ID, UPKA, UPKB)^{SSK} = H_1(ID, UPKA, UPKB)^s$ and $CertB = H_2(ID, UPKA, UPKB)^{SSK} = H_2(ID, UPKA, UPKB)^s$. In addition, the CA sends $Cert_{pair}$ to the user via a secure channel. Here, the user will securely store $Cert_{pair}$ to maintain its confidentiality. Next, the user can compute the two equations, namely $e(CertA, U) = e(H_1(ID, UPKA, UPKB), SPK)$ and $e(CertB, U) = e(H_2(ID, UPKA, UPKB), SPK)$, to ascertain the origin of the certificate pair $Cert_{pair}$. If the two equations hold true, $Cert_{pair}$ is confirmed to be from the CA; otherwise, it is not.
- **Encryption**: A sender inputs the system public parameters SPP, an identity ID, user public key pair UPK_{pair} and a message $M \in \{0, 1\}^\lambda$ to generate a ciphertext CT if $e(UPKA, V) = e(SPK, UPKB)$ holds. The specific details of the ciphertext $CT = (CT_1, CT_2, CT_3, CT_4)$ are shown as below:
  - $CT_1 = U^\alpha$, where $\alpha = H_5(M, \kappa)$ and $\kappa \in \{0, 1\}^l$ is chosen in random.
  - $CT_2 = U^\beta$, where $\beta \in Z_q^*$ is a random value.
  - $CT_3 = H_3(e(UPKA, H_1(ID, UPKA, UPKB)^\beta), CT_1, CT_2) \oplus (M \,||\, \kappa)$.
  - $CT_4 = H_4(M)^\alpha \cdot H_6(e(UPKA, H_2(ID, UPKA, UPKB)^\beta))$.
- **Decryption**: The receiver inputs the system public parameters SPP, the ciphertext CT, her/his user secret key USK and $Cert_{pair}$ to obtain the message M by performing the following steps:
  - Compute $CT_3 \oplus H_3(e(CT_2, CertA^x), CT_1, CT_2)$ to gain $M' \,||\, \kappa'$.

Compute $\alpha' = H_5(M', \kappa')$ and return the message M' as M if $CT_1 = U^{\alpha'}$ and $CT_4 = H_4(M)^{\alpha'} \cdot H_6(e(CT_2, CertB^x))$ both hold.

The process of gaining the message M can be seen in the following.

$$
\begin{aligned}
& CT_3 \oplus H_3(e(CT_2, CertA^x), CT_1, CT_2) \\
&= H_3(e(UPKA, H_1(ID, UPKA, UPKB)^\beta), \\
&\quad CT_1, CT_2) \oplus (M \,||\, \kappa) \oplus H_3(e(CT_2, \\
&\quad CertA^x), CT_1, CT_2) \\
&= H_3(e(U^{sx}, H_1(ID, UPKA, UPKB)^\beta), CT_1, \\
&\quad CT_2) \oplus (M \,||\, \kappa) \oplus H_3(e(U^\beta, CertA^x), \\
&\quad CT_1, CT_2) \\
&= H_3(e(U^{sx}, H_1(ID, UPKA, UPKB)^\beta), CT_1, \\
&\quad CT_2) \oplus (M \,||\, \kappa) \oplus H_3(e(U^\beta, H_1(ID, \\
&\quad UPKA, UPKB)^{sx}), CT_1, CT_2) \\
&= H_3(e(U, H_1(ID, UPKA, UPKB)^{sx\beta}), CT_1, \\
&\quad CT_2) \oplus (M \,||\, \kappa) \oplus H_3(e(U, H_1(ID, \\
&\quad UPKA, UPKB)^{sx\beta}), CT_1, CT_2) \\
&= (M \,||\, \kappa).
\end{aligned}
$$

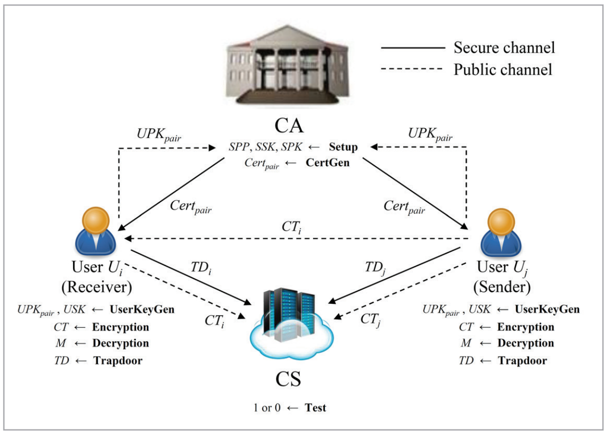
- **Trapdoor**: The user inputs system public parameters SPP, her/his $Cert_{pair}$ and user secret key USK to obtain a trapdoor $TD = CertB^{USK} = H_2(ID, UPKA, UPKB)^{sx}$. Then, the user sends TD to the CS via a secure channel. Here, the CS will securely store TD to maintain its confidentiality. Next, the CS can compute the equation, namely $e(TD, U) = e(H_2(ID, UPKA, UPKB), UPKA)$, to ascertain the origin of the trapdoor TD. If the equation holds true, TD is confirmed to be from the user; otherwise, it is not.
- **Test**: For any two users U_i and U_j, the CS inputs the system public parameters SPP, two ciphertexts CT_i, CT_j and two trapdoors TD_i, TD_j, where $CT_i = (CT_{i1}, CT_{i2}, CT_{i3}, CT_{i4})$ and $CT_j = (CT_{j1}, CT_{j2}, CT_{j3}, CT_{j4})$ to return 1 or 0 by performing the following steps.
  - Compute T_i and T_j as below.
    - $T_i = \dfrac{CT_{i4}}{H_6(e(CT_{i2}, TD_i))}$

$$
\begin{aligned}
&= \frac{H_4(M_i)^{\alpha_i} \cdot H_6(e(UPKA_i, H_2(ID_i, UPKA_i, UPKB_i))^{\beta_i})}{H_6(e(U^{\beta_i}, CertB_i^{USK}))} \\
&= \frac{H_4(M_i)^{\alpha_i} \cdot H_6(e(SPK^{x_i}, H_2(ID_i, UPKA_i, UPKB_i))^{\beta_i})}{H_6(e(U^{\beta_i}, H_2(ID_i, UPKA_i, UPKB_i)^{sx_i}))} \\
&= \frac{H_4(M_i)^{\alpha_i} \cdot H_6(e(U^{sx_i}, H_2(ID_i, UPKA_i, UPKB_i))^{\beta_i})}{H_6(e(U^{sx_i}, H_2(ID_i, UPKA_i, UPKB_i))^{\beta_i})} \\
&= H_4(M_i)^{H_5(M_i, \kappa_i)}
\end{aligned}
$$

- $$\mathrm{T}_j = \frac{CT_{j4}}{H_6(e(CT_{j2}, TD_j))}$$
  $$= \frac{H_4(M_j)^{\alpha_j} \cdot H_6(e(UPKA_j, H_2(ID_j, UPKA_j, UPKB_j))^{\beta_j})}{H_6(e(U^{\beta_j}, CertB_j{}^{USK}))} =$$
  $$\frac{H_4(M_j)^{\alpha_j} \cdot H_6(e(SPK^{x_j}, H_2(ID_j, UPKA_j, UPKB_j))^{\beta_j})}{H_6(e(U^{\beta_j}, H_2(ID_j, UPKA_j, UPKB_j)^{sx_j}))}$$
  $$= \frac{H_4(M_j)^{\alpha_j} \cdot H_6(e(U^{sx_j}, H_2(ID_j, UPKA_j, UPKB_j))^{\beta_j})}{H_6(e(U^{sx_j}, H_2(ID_j, UPKA_j, UPKB_j))^{\beta_j})}$$
  $$= H_4(M_j)^{H_5(M_j, \kappa_j)}$$
- Calculate $e(CT_{i1}, T_j)$ and $e(CT_{j1}, T_i)$ as below.
  - $$e(CT_{i1}, T_j) = e(U^{H_5(M_i, \kappa_i)}, H_4(M_j)^{H_5(M_j, \kappa_j)})$$
    $$= e(U, H_4(M_j))^{H_5(M_i, \kappa_i) \cdot H_5(M_j, \kappa_j)}$$
  - $$e(CT_{j1}, T_i) = e(U^{H_5(M_j, \kappa_j)}, H_4(M_i)^{H_5(M_i, \kappa_i)})$$
    $$= e(U, H_4(M_i))^{H_5(M_i, \kappa_i) \cdot H_5(M_j, \kappa_j)}$$
- Return 1 if $e(CT_{i1}, T_j) = e(CT_{j1}, T_i)$. Otherwise, return 0.

Figure 4

Visual representation of the CBEET scheme



5. Security Analysis

As mentioned in Section 3, four types of attackers were defined. Meanwhile, two security games $G_{\text{CBEET-IND-CCA}}$ and $G_{\text{CBEET-OW-CCA}}$ respectively were used to model indistinguishabilty under chosen ciphertext attacks and one-wayness under chosen ciphertext attacks. As considering these security notions, four theorems are given to demonstrate that the proposed CB-PKEET scheme is secure.

Theorem 1. Assume that six hash functions H_i, for $i \in [1, 6]$, are random oracles and $\mathcal{A}_1$ is a Type I adversary against the CB-PKEET scheme with advantage ϵ in the security game $G_{\text{CBEET-IND-CCA}}$. Then, there is an algorithms $\mathcal{C}$ to solve the BDH problem with advantage

$$\epsilon' \geq (1/q_{H_3}) \, [\epsilon/e(q_{cer} + q_{trap} + 1) - q_d/q - q_{H_6}/q],$$

where q_{H_3}, q_{H_6}, q_{cer}, q_{trap} and q_d respectively are query times to random oracle H_3, random oracle H_6, certification query, trapdoor query and decryption query.

Proof. An algorithm $\mathcal{C}$ is given an instance of the BDH problem: $(\mathcal{G}, U, U^a, U^c, V, V^a, V^b)$ where $\mathcal{G} = (q, G_1, G_2, G_T, \hat{e})$. Let $D = \hat{e}(U, V)^{abc} \in G_T$ be the solution of the BDH problem. The algorithm $\mathcal{C}$ simulates a challenger to find D by interacting with $\mathcal{A}_1$ in the following security game $G_{\text{CBEET-IND-CCA}}$.

- **Setup.** The challenger $\mathcal{C}$ generates the system public parameter $SPP = (\mathcal{G}, U, V, SPK, H_1, H_2, H_3, H_4, H_5, H_6)$ by setting $SPK = U^a$. Then, the system public parameter SPP is sent to $\mathcal{A}_1$. Here, $H_1, H_2, \ldots, H_6$ are hash functions as random oracles. Because $\mathcal{C}$'s responses to queries to these random oracles issued from $\mathcal{A}_1$ must be consistent, $\mathcal{C}$ must maintain the several lists, namely $L_{H1}, L_{H2}, \ldots, L_{H6}, L_{Key}$ which are defined in phase 1 below.
- **Phase 1.** The adversary $\mathcal{A}_1$ may issue the following queries.
  - H_1 *query*. When receiving this query with an identity ID and the user public key pair $UPK_{pair} = (UPKA, UPKB)$, the challenger $\mathcal{C}$ uses them to search the list L_{H1} of the form $[ID, UPKA, UPKB, \sigma, cn]$.
    - If $(ID, UPKA, UPKB)$ appears on the list L_{H1}, $\mathcal{C}$ uses the corresponding σ and cn to return V^{σ} if $cn = 0$ or $V^{b\sigma}$ if $cn = 1$.
    - Otherwise, $\mathcal{C}$ uses the identity ID to perform *user public key query* to gain σ and cn, and $\mathcal{C}$ records them into the list L_{H1}.
  - H_2 *query*. When receiving this query with an identity ID and the user public key pair $UPK_{pair} = (UPKA, UPKB)$, the challenger $\mathcal{C}$ uses them to search the list L_{H2} of the form $[ID, UPKA, UPKB, \tau, cn]$.
    - If $(ID, UPKA, UPKB)$ appears on the list L_{H2}, $\mathcal{C}$ uses the corresponding τ and cn to return V^{τ} if $cn = 0$ or $V^{b\tau}$ if $cn = 1$.

  - Otherwise, $\mathcal{C}$ uses the identity ID to perform *user public key query* to gain τ and cn, and $\mathcal{C}$ records them into the list L_{H2}.
- H_3 *query*. When receiving this query with a value $\mu \in G_T$ and two points $CT_1, CT_2 \in G_1$, the challenger $\mathcal{C}$ uses them to search the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$.
  - If (μ, CT_1, CT_2) appears on the list L_{H3}, $\mathcal{C}$ returns the corresponding φ.
  - Otherwise, $\mathcal{C}$ randomly chooses a value $\varphi \in \{0, 1\}^{\lambda+l}$ as answer to $\mathcal{A}_1$, and records $[\mu, CT_1, CT_2, \varphi]$ into the list L_{H3}.
- H_4 *query*. When receiving this query with a value $M \in \{0, 1\}^{\lambda}$, the challenger $\mathcal{C}$ uses it to search the list L_{H4} of the form $[M, \gamma]$.
  - If M appears on the list L_{H4}, $\mathcal{C}$ returns the corresponding γ.
  - Otherwise, $\mathcal{C}$ randomly chooses a point $\gamma \in G_2$ as answer to $\mathcal{A}_1$, and records $[M, \gamma]$ into the list L_{H4}.
- H_5 *query*. When receiving this query with two values $M \{0, 1\}^{\lambda}$, $\kappa \in \{0, 1\}^{l}$, the challenger $\mathcal{C}$ uses them to search the list L_{H5} of the form $[M, \kappa, \theta]$.
  - If (M, κ) appears on the list L_{H5}, $\mathcal{C}$ returns the corresponding θ.
  - Otherwise, $\mathcal{C}$ randomly chooses a value $\theta \in Z_q^*$ as answer to $\mathcal{A}_1$, and records $[M, \gamma]$ into the list L_{H5}.
- H_6 *query*. When receiving this query with a value $\zeta \in G_T$, the challenger $\mathcal{C}$ uses it to search the list L_{H6} of the form $[\zeta, \eta]$.
  - If ζ appears on the list L_{H6}, $\mathcal{C}$ returns the corresponding η.
  - Otherwise, $\mathcal{C}$ randomly chooses a point $\eta \in G_2$ as answer to $\mathcal{A}_1$, and records $[\zeta, \eta]$ into the list L_{H6}.
- *User public key query*. When receiving this query with an identity ID, the challenger $\mathcal{C}$ randomly chooses two values $\sigma \in Z_q^*, \tau \in Z_q^*$ and a coin $cn \in \{0, 1\}$. Then $\mathcal{C}$ respectively records two tuples $[ID, UPKA, UPKB, \sigma, cn]$ and $[ID, UPKA, UPKB, \tau, cn]$ into the lists L_{H1} and L_{H2} by doing the following setting.
  - If $cn = 0$ with the probability $\Pr[cn = 0] = \upsilon$, $\mathcal{C}$ runs the UserKeyGen algorithm to obtain the user secret key $USK = x$. Then, $\mathcal{C}$ uses it to calculate user public key $UPK_{pair} = (UPKA, UPKB) = (SPK^x, V^x)$. Further, the certificate pair $Cert_{pair} = (CertA, CertB) = (V^{a\sigma}, V^{a\tau})$ can be obtained by performing the CertGen algorithm. Finally, $\mathcal{C}$ records $[ID, USK, UPKA, UPKB, CertA, CertB, 0]$ into the list L_{Key}, and returns $UPK_{pair} = (UPKA, UPKB)$ to $\mathcal{A}_1$.
  - Otherwise, $\mathcal{C}$ runs the UserKeyGen algorithm to obtain the user secret key $USK = x$. Then, $\mathcal{C}$ uses it to calculate user public key $UPK_{pair} = (UPKA, UPKB) = (SPK^x, V^x)$. Finally, $\mathcal{C}$ records $[ID, USK, UPKA, UPKB, \text{-}, \text{-}, 1]$ into the list L_{key}, and returns $UPK_{pair} = (UPKA, UPKB)$ to $\mathcal{A}_1$.
- *User secret key query*. When receiving this query with an identity ID, the challenger C uses it to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$.
  - If ID appears on the list L_{key}, $\mathcal{C}$ returns the corresponding USK.
  - Otherwise, $\mathcal{C}$ uses the identity ID to perform user public key query to gain USK, and $\mathcal{C}$ returns it.
- *User Public key replace query*. When receiving this query with an identity ID and a new user public key pair $UPK_{pair}' = (UPKA', UPKB')$, the challenger $\mathcal{C}$ checks whether $\hat{e}(UPKA', V) = \hat{e}(SPK, UPKB')$ or not.
  - If $\hat{e}(UPKA', V) = \hat{e}(SPK, UPKB')$ holds, the original user public key $UPK_{pair} = (UPKA, UPKB)$ related to the identity ID will be replaced with UPK_{pair}'.
  - Otherwise, $\mathcal{C}$ returns $\perp$ *to* $\mathcal{A}_1$.
- *Certification query*. When receiving this query with an identity ID and user public key $UPK_{pair} = (UPKA, UPKB)$, the challenger $\mathcal{C}$ uses them to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$.
  - If $(ID, UPKA, UPKB)$ appears on the list L_{key}, $\mathcal{C}$ returns the corresponding certificate pair $Cert_{pair} = (CertA, CertB)$ if $cn = 0$ or aborts this game if $cn = 1$.
  - Otherwise, $\mathcal{C}$ performs user public key query to record the corresponding information. Then, $\mathcal{C}$ runs the query again to return $Cert_{pair} = (CertA, CertB)$ or abort this game.

- *Decryption query*. When receiving this query with an identity *ID* and ciphertext $CT = (CT_1, CT_2, CT_3, CT_4)$, the challenger $\mathcal{C}$ uses them to search the list L_{key} of the form [*ID, USK, UPKA, UPKB, CertA, CertB, cn*].
  - If ID appears on the list L_{key} and $cn = 0$, $\mathcal{C}$ uses the corresponding *USK*, *CertA* and *CertB* to perform the Decryption algorithm. Then, $\mathcal{C}$ returns the output.
  - Otherwise, $\mathcal{C}$ uses (CT_1, CT_2) to search the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$. If (CT_1, CT_2) can be found, $\mathcal{C}$ calculates $M' \| \kappa' = CT_3 \oplus \varphi$ by using the corresponding φ. Then, $M' \| \kappa'$ is used to search the list L_{H4} of the form $[M, \gamma]$ and the list L_{H5} of the form $[M, \kappa, \theta]$. If η can be found on the list L_{H6} of the form $[\zeta, \eta]$ such that $CT_4 = \gamma \cdot \eta$ holds, $\mathcal{C}$ calculates $CT_1' = U^{\theta}$. If $CT_1' = CT_1$, $\mathcal{C}$ returns M' to $\mathcal{A}_1$. Otherwise, returns $\perp$ to $\mathcal{A}_1$.
- *Trapdoor query*. When receiving this query with an identity *ID*, the challenger $\mathcal{C}$ uses it to search the list L_{key} of the form [*ID, USK, UPKA, UPKB, CertA, CertB, cn*].
  - If *ID* appears on the list L_{key}, $\mathcal{C}$ returns the corresponding trapdoor $TD = CertB^{USK}$ if $cn = 0$ or aborts this game if $cn = 1$.
  - Otherwise, $\mathcal{C}$ performs user public key query to record the corresponding information. Then, $\mathcal{C}$ runs the query again to return *TD* or abort this game.

– **Challenge.** The adversary $\mathcal{A}_1$ sends a message pair (M_0^*, M_1^*) and a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ uses ID^* to search the list L_{key} of the form [*ID, USK, UPKA, UPKB, CertA, CertB, cn*]. If $cn = 0$, $\mathcal{C}$ aborts this game. Otherwise, $\mathcal{C}$ pick a random value $rv \in \{0, 1\}$, and uses M_{rv}^* and κ to issue H_5 *query* to gain θ, where the value $\kappa \in \{0, 1\}^l$ is chosen in random. Further, $\mathcal{C}$ sets $CT_1^* = U^{\theta}$, $CT_2^* = U^c$, $CT_3^* \in \{0, 1\}^{\lambda+l}$ and $CT_4^* \in G_2$. Here, CT_3^* and CT_4^* respectively are a random value in $\{0, 1\}^{\lambda+l}$ and a random point in G_2. Finally, $\mathcal{C}$ runs the target ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, CT_4^*)$.

– **Phase 2.** The adversary $\mathcal{A}_1$ can continue to issue the same queries as in phase 1.

– **Guess.** The adversary $\mathcal{A}_1$ sends a value $rv' \in \{0, 1\}$ as the answer to guess. $\mathcal{A}_1$ wins this game if $rv' = rv$. $\mathcal{C}$ randomly selects a tuple $[\mu^*, CT_1^*, CT_2^*, \varphi^*]$ from the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$ to gain μ^*. Then, $D = (\mu^*)^{(x^*\sigma^*)^{\wedge}-1}$ is outputted as the solution of the BDH problem.

Analysis. We first discuss the simulations of hash functions, namely $H_1, H_2, \ldots, H_6$. Obviously, we can say that the hash functions H_1, H_2, H_4, and H_5 as the random oracles are perfect simulations since the inputs and outputs of the random oracles are independent of the solution of the BDH problem. Assume that $EventH_3^*$ and $EventH_6^*$ are two events of issuing the H_3 query with $(e(U, V)^{abcx^*\sigma^*}, CT_1^*, CT_2^*)$ and H_6 query with $(e(U, V)^{abcx^*\tau^*})$, respectively. Here, we say that the hash functions H_3 and H_6 as the random oracles are perfect simulations if both two events $EventH_3^*$ and $EventH_6^*$ did not occur. We then discuss the simulations of the *decryption query*. We denote *EventDecFail* as the event that the ciphertext is valid, and the challenger $\mathcal{C}$ is unable to decrypt it. The probability of this event is $\Pr[EventDecFail] \leq q_d/q$.

Moreover, we denote $Event = (EventH_3^* \vee EventH_6^* \vee EventDecFail) | \neg\, EventAbor$ as the event that this game will not be aborted, where *EventAbort* is the event that the challenger $\mathcal{C}$ aborts this game. We can obtain probability $\Pr[rv = rv' \mid \neg\, Event] \leq 1/2$ if the event *Event* does not occur. Further, we get

$$\begin{aligned}\Pr[rv = rv'] &= \Pr[rv = rv' | Event]\Pr[Event] \\ &\quad + \Pr[rv = rv' | \neg\, Event]\Pr[\neg Event] \\ &\leq \Pr[Event] + (1/2)\cdot\Pr[\neg Event] \\ &= \Pr[Event] + (1/2)\cdot(1 - \Pr[Event]) \\ &= (1/2)\cdot\Pr[Event] + 1/2.\end{aligned}$$

According to the sense of ϵ, we have $\epsilon = \Pr[rv = rv'] - 1/2$. Hence, we obtain $\epsilon = \Pr[rv = rv'] - 1/2 \leq \Pr[Event] \leq (\Pr[EventH_3^*] + \Pr[EventH_6^*] + \Pr[EventDecFail]) / \Pr[\neg EventAbor]$. By this inequality, we have $\Pr[EventH_3^*] \geq \epsilon \cdot \Pr[\neg EventAbor] - \Pr[EventDecFail] - \Pr[EventH_6^*]$. Since $\Pr[\neg EventAbor] = \upsilon^{q_{cer} + q_{trap}}(1 - \upsilon)$, we can gain $\Pr[\neg EventAbor] \geq 1/ e(q_{cer} + q_{trap} + 1)$ when $\upsilon = 1 - 1/(q_{cer} + q_{trap} + 1)$. We then have $\Pr[EventH_3^*] \geq \epsilon/e(q_{cer} + q_{trap} + 1) - q_d/q - q_{H_6}/q$.

If the event $EventH_3^*$ occurs, the adversary A_1 can know the the the target ciphertext CT^* is invalid. $H_3(e(P, Q)^{abcx^*\sigma^*}, CT_1^*, CT_2^*)$ has been recorded in the list L_{H3}. We can say that the challenger $\mathcal{C}$ wins this game if the correct element was chosen in the list L_{H3}. Therefore, the challenger $\mathcal{C}$ can solve the BDH problem with advantage

$\epsilon' \geq (1/q_{H_3})\Pr[EventH_3^*]$

$\quad \geq (1/q_{H_3})\cdot[\epsilon/e(q_{cer} + q_{trap} + 1) - q_d/q - q_{H_6}/q].$

Theorem 2. Assume that six hash functions H_i, for $i \in [1, 6]$, are random oracles and $\mathcal{A}_2$ is a Type II adversary against the CB-PKEET scheme with advantage ϵ in the security game $G_{\text{CBEET-IND-CCA}}$. Then, there is an algorithms $\mathcal{C}$ to solve the BDH problem with advantage

$\epsilon' \geq (1/q_{H_3})\,[\epsilon/e(q_{trap} + 1) - q_d/q - q_{H_6}/q],$

where q_{H_3}, q_{H_6}, q_{trap} and q_d respectively are query times to random oracle H_3, random oracle H_6, trapdoor queries and decryption queries.

Proof. An algorithm $\mathcal{C}$ is given an instance of the BDH problem: $(\mathcal{G}, U, U^a, U^c, V, V^a, V^b)$ where $\mathcal{G} = (q, G_1, G_2, G_T, \hat{e})$. Let $D = \hat{e}(U, V)^{abc} \in G_T$ be the solution of the BDH problem. The algorithm $\mathcal{C}$ simulates a challenger to find D by interacting with $\mathcal{A}_2$ in the following security game $G_{\text{CBEET-IND-CCA}}$.

- **Setup.** The challenger $\mathcal{C}$ generates the system public parameter $SPP = (\mathcal{G}, U, V, SPK, H_1, H_2, H_3, H_4, H_5, H_6)$ by setting $SPK = U^s$, where $s \in Z_q^*$ is random value as the system secret key SSK. Then, the system public parameter SPP is sent to $\mathcal{A}_2$. Here, $H_1, H_2,..., H_6$ are hash functions as random oracles. Because $\mathcal{C}$'s responses to queries to these random oracles issued from $\mathcal{A}_2$ must be consistent, $\mathcal{C}$ must maintain the several lists, namely L_{H1}, L_{H2}, ..., L_{H6}, L_{Key} which are defined in phase 1 below.
- **Phase 1.** The adversary $\mathcal{A}_2$ may issue the following queries.
  - *H_1-H_6 queries.* The response is similar to the proof of Theorem 1.
  - *User public key query.* When receiving this query with an identity ID, the challenger $\mathcal{C}$ randomly chooses two values $\sigma \in Z_q^*$, $\tau \in Z_q^*$ and a coin $cn \in \{0, 1\}$. Then $\mathcal{C}$ respectively records two tuples $[ID, UPKA, UPKB, \sigma, cn]$ and $[ID, UPKA, UPKB, \tau, cn]$ into the lists L_{H1} and L_{H2} by doing the following setting.
    - If $cn = 0$ with the probability $\Pr[cn = 0] = \upsilon$, $\mathcal{C}$ runs the UserKeyGen algorithm to obtain the user secret key $USK = x$. Then, $\mathcal{C}$ uses it to calculate user public key $UPK_{pair} = (UPKA, UPKB) = (SPK^x, V^x)$. Further, the certificate pair $Cert_{pair} = (CertA, CertB) = (V^{s\sigma}, V^{s\tau})$ can be obtained by performing the CertGen algorithm. Finally, $\mathcal{C}$ records $[ID, USK, UPKA, UPKB, CertA, CertB, 0]$ into the list L_{Key}, and returns $UPK_{pair} = (UPKA, UPKB)$ to $\mathcal{A}_2$.
    - Otherwise, $\mathcal{C}$ randomly select a value $x' \in Z_q^*$ as the user secret key USK'. Then, $\mathcal{C}$ uses it to calculate user public key $UPK_{pair} = (UPKA, UPKB) = (SPK^{ax'}, V^{ax'})$. Further, the certificate pair $Cert_{pair} = (CertA, CertB) = (V^{sb\sigma}, V^{sb\tau})$ can be obtained by owning system secret key $SSK = s$. Finally, $\mathcal{C}$ records $[ID, USK', UPKA, UPKB, CertA, CertB, 1]$ into the list L_{key}, and returns $UPK_{pair} = (UPKA, UPKB)$ to $\mathcal{A}_2$. Note that the user secret key $USK = x$ is treated as ax'.
  - *User secret key query.* The response is similar to the proof of Theorem 1.
  - *Decryption query.* The response is similar to the proof of Theorem 1.
  - *Trapdoor query.* The response is similar to the proof of Theorem 1.
- **Challenge.** The adversary $\mathcal{A}_2$ sends a message pair (M_0^*, M_1^*) and a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ uses ID^* to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$. If $cn = 0$, $\mathcal{C}$ aborts this game. Otherwise, $\mathcal{C}$ pick a random value $rv \in \{0, 1\}$, and uses M_{rv}^* and κ to issue H_5 *query* to gain θ, where the value $\kappa \in \{0, 1\}^l$ is chosen in random. Further, $\mathcal{C}$ sets $CT_1^* = U^\theta$, $CT_2^* = U^c$, $CT_3^* \in \{0, 1\}^{\lambda+l}$ and $CT_4^* \in G_2$. Here, CT_3^* and CT_4^* respectively are a random value in $\{0, 1\}^{\lambda+l}$ and a random point in G_2. Finally, $\mathcal{C}$ runs the target ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, CT_4^*)$.
- **Phase 2.** The adversary $\mathcal{A}_2$ can continue to issue the same queries as in phase 1.

Guess. The adversary $\mathcal{A}_2$ sends a value $rv' \in \{0, 1\}$ as the answer to guess. $\mathcal{A}_2$ wins this game if $rv' = rv$. $\mathcal{C}$ randomly selects a tuple $[\mu^*, CT_1^*, CT_2^*, \varphi^*]$ from the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$ to gain μ^*. Then, $D = \mu^*/\hat{e}(U, V)^{s^2x^*\sigma^*}$ is outputted as the solution of the BDH problem.

Analysis. We first discuss the simulations of hash functions, namely $H_1, H_2,..., H_6$. Obviously, we can say that the hash functions H_1, H_2, H_4, and H_5 as the random oracles are perfect simulations since the inputs and outputs of the random oracles are independent of the solution of the BDH problem. Assume that $EventH_3^*$ and $EventH_6^*$ are two events of issuing the H_3 query with $(\hat{e}(U, V)^{abcs^2x^*\sigma^*}, CT_1^*, CT_2^*)$ and H_6 query with $(\hat{e}(U, V)^{abcs^2x^*\tau^*})$, respectively. Here, we say that the hash functions H_3 and H_6 as the random oracles are perfect simulations if both two events $EventH_3^*$ and $EventH_6^*$ did not occur. We then discuss the simulations of the *decryption query*. We denote *EventDecFail* as the event that the ciphertext is valid, and the challenger $\mathcal{C}$ is unable to decrypt it. The probability of this event is $\Pr[EventDecFail] \le q_d/q$.

Moreover, we denote $Event = (EventH_3^* \lor EventH_6^* \lor EventDecFail)|\neg\ EventAbor$ as the event that this game will not be aborted, where *EventAbort* is the event that the challenger $\mathcal{C}$ aborts this game. We can obtain probability $\Pr[rv = rv' |\neg\ Event] \le 1/2$ if the event *Event* does not occur. Further, we get

$$\begin{aligned}\Pr[rv = rv'] &= \Pr[rv = rv'|Event]\Pr[Event]\\ &\quad + \Pr[rv = rv'|\neg\ Event]\Pr[\neg Event]\\ &\le \Pr[Event] + (1/2)\cdot\Pr[\neg Event]\\ &= \Pr[Event] + (1/2)\cdot(1 - \Pr[Event])\\ &= (1/2)\cdot\Pr[Event] + 1/2.\end{aligned}$$

According to the sense of ϵ, we have $\epsilon = \Pr[rv = rv'] - 1/2$. Hence, we obtain $\epsilon = \Pr[rv = rv'] - 1/2 \le \Pr[Event] \le (\Pr[EventH_3^*] + \Pr[EventH_6^*] + \Pr[EventDecFail]) / \Pr[\neg EventAbor]$. By this inequality, we have $\Pr[EventH_3^*] \ge \epsilon\cdot\Pr[\neg EventAbor] - \Pr[EventDecFail] - \Pr[EventH_6^*]$. Since $\Pr[\neg EventAbor] = \upsilon^{q_{trap}}(1 - \upsilon)$, we can gain $\Pr[\neg EventAbor] \ge 1/e(q_{trap} + 1)$ when $\upsilon = 1 - 1/(q_{trap} + 1)$. We then have $\Pr[EventH_3^*] \ge \epsilon/e(q_{trap} + 1) - q_d/q - q_{H_6}/q$.

If the event $EventH_3^*$ occurs, the adversary A_2 can know the the the target ciphertext CT^* is invalid. $H_3(\hat{e}(U, V)^{abcs^2x^*\sigma^*}, CT_1^*, CT_2^*)$ has been recorded in the list L_{H3}. We can say that the challenger $\mathcal{C}$ wins this game if the correct element was chosen in the list L_{H3}. Therefore, the challenger $\mathcal{C}$ can solve the BDH problem with advantage

$$\begin{aligned}\epsilon' &\ge (1/q_{H_3})\Pr[EventH_3^*]\\ &\ge (1/q_{H_3})\cdot[\epsilon/e(q_{trap} + 1) - q_d/q - q_{H_6}/q].\end{aligned}$$

Theorem 3. Assume that six hash functions H_i, for $i \in [1, 6]$, are random oracles and $\mathcal{A}_3$ is a Type III adversary against the CB-PKEET scheme with advantage ϵ in the security game $G_{\text{CBEET-OW-CCA}}$. Then, there is an algorithms $\mathcal{C}$ to solve the BDH problem with advantage

$$\epsilon' \ge (1/q_{H_3})\,[(\epsilon - 1/2^{\lambda})/\,e(q_{cer} + 1) - q_d/q],$$

where q_{H_3}, q_{cer} and q_d respectively are query times to random oracle H_3, certification queries and decryption queries.

Proof. An algorithm $\mathcal{C}$ is given an instance of the BDH problem: $(\mathcal{G}, U, U^a, U^c, V, V^a, V^b)$ where $\mathcal{G} = (q, G_1, G_2, G_T, \hat{e})$. Let $D = \hat{e}(U, V)^{abc} \in G_T$ be the solution of the BDH problem. The algorithm $\mathcal{C}$ simulates a challenger to find D by interacting with $\mathcal{A}_3$ in the following security game $G_{\text{CBEET-OW-CCA}}$.

- **Setup.** The challenger $\mathcal{C}$ generates the system public parameter $SPP = (\mathcal{G}, U, V, SPK, H_1, H_2, H_3, H_4, H_5, H_6)$ by setting $SPK = U^a$. Then, the system public parameter SPP is sent to $\mathcal{A}_3$. Here, $H_1, H_2, \ldots, H_6$ are hash functions as random oracles. Because $\mathcal{C}$'s responses to queries to these random oracles issued from $\mathcal{A}_3$ must be consistent, $\mathcal{C}$ must maintain the several lists, namely $L_{H1}, L_{H2}, \ldots, L_{H6}, L_{Key}$ which are defined in phase 1 below.
- **Phase 1.** The adversary $\mathcal{A}_3$ may issue the following queries.
  - H_1 *query*. The response is similar to the proof of Theorem 1.
  - H_2 *query*. When receiving this query with an identity ID and the user public key pair $UPK_{pair} = (UPKA, UPKB)$, the challenger $\mathcal{C}$ uses them to search the list L_{H2} of the form $[ID, UPKA, UPKB, \tau, cn]$.
    - If $(ID, UPKA, UPKB)$ appears on the list L_{H2}, $\mathcal{C}$ uses the corresponding τ to return V^{τ}.
    - Otherwise, $\mathcal{C}$ uses the identity ID to perform *user public key query* to gain τ, and $\mathcal{C}$ records it into the list L_{H2}.
  - H_3-H_6 *queries*. The response is similar to the proof of Theorem 1.
  - *User public key query*. The response is similar to the proof of Theorem 1.
  - *User secret key query*. The response is similar to the proof of Theorem 1.
  - *Certification query*. The response is similar to the proof of Theorem 1.
  - *Decryption query*. When receiving this query with an identity ID and ciphertext $CT = (CT_1, CT_2, CT_3, CT_4)$, the challenger $\mathcal{C}$ uses them to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$.
    - If ID appears on the list L_{key} and $cn = 0$, $\mathcal{C}$ uses the corresponding USK, $CertA$ and $CertB$ to perform the Decryption algorithm. Then, $\mathcal{C}$ returns the output.

   - Otherwise, $\mathcal{C}$ uses (CT_1, CT_2) to search the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$. If (CT_1, CT_2) can be found, $\mathcal{C}$ calculates $M' \| \kappa' = CT_3 \bigoplus \varphi$ by using the corresponding φ. Then, $M' \| \kappa'$ is used to search the list L_{H4} of the form $[M, \gamma]$ and the list L_{H5} of the form $[M, \kappa, \theta]$. In addition, $\mathcal{C}$ uses ID to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$ to calculate $CertB^{USK} = V^{\alpha\tau x}$. If η can be found on the list L_{H6} of the form $[e(CT_2, CertB^{USK}), \eta]$ such that $CT_4 = \gamma \cdot \eta$ holds, $\mathcal{C}$ calculates $CT_1' = U^{\theta}$. If $CT_1' = CT_1$, $\mathcal{C}$ returns M' *to* $\mathcal{A}_3$. Otherwise, returns $\perp$ to $\mathcal{A}_3$.
- *Trapdoor query*. When receiving this query with an identity ID, the challenger $\mathcal{C}$ uses it to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$.
   - If ID appears on the list L_{key}, $\mathcal{C}$ returns the corresponding trapdoor $TD = CertB^{USK}$.
   - Otherwise, $\mathcal{C}$ performs user public key query to record the corresponding information. Then, $\mathcal{C}$ runs the query again to return TD.

Challenge. The adversary $\mathcal{A}_3$ sends a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ uses ID^* to search the list L_{key} of the form $[ID, USK, UPKA, UPKB, CertA, CertB, cn]$. If $cn = 0$, $\mathcal{C}$ aborts this game. Otherwise, $\mathcal{C}$ picks a random message $M^* \in \{0, 1\}^{\lambda}$ and a value $\kappa \in \{0, 1\}^{l}$, and then uses M^* and κ to issue H_5 *query* to gain θ. In addition, $\mathcal{C}$ uses M^* and $\hat{e}(U^c, V^{\alpha\tau^* x^*})$ to issue H_4 *query* and H_6 *query* to gain γ and η. Further, $\mathcal{C}$ sets $CT_1^* = U^{\theta}$, $CT_2^* = U^c$, $CT_3^* \in \{0, 1\}^{\lambda+l}$ and $CT_4^* = \gamma^{\theta} \cdot \eta$. Here, CT_3^* is a random value in $\{0, 1\}^{\lambda+l}$. Finally, $\mathcal{C}$ runs the target ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, CT_4^*)$.

Phase 2. The adversary $\mathcal{A}_3$ can continue to issue the same queries as in phase 1.

Guess. The adversary $\mathcal{A}_3$ sends a message $M' \in \{0, 1\}^{\lambda}$ as the answer to guess. $\mathcal{A}_3$ wins this game if $M' = M^*$. $\mathcal{C}$ randomly selects a tuple $[\mu^*, CT_1^*, CT_2^*, \varphi^*]$ from the list L_{H3} of the form $[\mu, CT_1, CT_2, \varphi]$ to gain μ^*. Then, $D = (\mu^*)^{(x^* \sigma^*)\hat{}-1}$ is outputted as the solution of the BDH problem.

Analysis. We first discuss the simulations of hash functions, namely $H_1, H_2, \ldots, H_6$. Obviously, we can say that the hash functions H_1, H_2, H_4, H_5 and H_6 as the random oracles are perfect simulations since the inputs and outputs of the random oracles are independent of the solution of the BDH problem. Assume that $EventH_3^*$ is a events of issuing the H_3 query with $(\hat{e}(U, V)^{abcx^*\sigma^*}, CT_1^*, CT_2^*)$. Here, we say that the hash functions H_3 as the random oracles is perfect simulations if events $EventH_3^*$ did not occur. We then discuss the simulations of the *decryption query*. We denote *EventDecFail* as the event that the ciphertext is valid, and the challenger $\mathcal{C}$ is unable to decrypt it. The probability of this event is $\Pr[EventDecFail] \leq q_d/q$.

Moreover, we denote $Event = (EventH_3^* \vee EventDecFail) | \neg EventAbor$ as the event that this game will not be aborted, where *EventAbort* is the event that the challenger $\mathcal{C}$ aborts this game. We can obtain probability $\Pr[M' = M^* | \neg Event] \leq 1/2^{\lambda}$ if the event *Event* does not occur. Further, we get

$$\begin{aligned}\Pr[rv = rv'] &= \Pr[M' = M^* | Event]\Pr[Event] \\ &\quad + \Pr[rv = rv' | \neg Event]\Pr[\neg Event] \\ &\leq \Pr[Event] + (1/2^{\lambda}) \cdot \Pr[\neg Event] \\ &= \Pr[Event] + (1/2^{\lambda}) \cdot (1 - \Pr[Event]) \\ &= (1/2^{\lambda}) \cdot \Pr[Event] + 1/2^{\lambda}.\end{aligned}$$

According to the sense of ϵ, we have $\epsilon = \Pr[M' = M^*] - 1/2^{\lambda}$. Hence, we obtain $\epsilon = \Pr[M' = M^*] - 1/2^{\lambda} \leq \Pr[Event] \leq (\Pr[EventH_3^*] + \Pr[EventDecFail]) / \Pr[\neg EventAbor]$. By this inequality, we have $\Pr[EventH_3^*] \geq (\epsilon - 1/2^{\lambda}) \cdot \Pr[\neg EventAbor] - \Pr[EventDecFail]$. Since $\Pr[\neg EventAbor] = \upsilon^{q_{cer}}(1 - \upsilon)$, we can gain $\Pr[\neg EventAbor] \geq 1/e(q_{cer} + 1)$ when $\upsilon = 1 - 1/(q_{cer} + 1)$. We then have $\Pr[EventH_3^*] \geq (\epsilon - 1/2^{\lambda})/e(q_{cer} + 1) - q_d/q$.

If the event $EventH_3^*$ occurs, the adversary A_3 can know the the the target ciphertext CT^* is invalid. $H_3(e(U, V)^{abcx^*\sigma^*}, CT_1^*, CT_2^*)$ has been recorded in the list L_{H3}. We can say that the challenger $\mathcal{C}$ wins this game if the correct element was chosen in the list L_{H3}. Therefore, the challenger $\mathcal{C}$ can solve the BDH problem with advantage

$$\begin{aligned}\epsilon' &\geq (1/q_{H_3})\Pr[EventH_3^*] \\ &\geq (1/q_{H_3}) \cdot [(\epsilon - 1/2^{\lambda})/e(q_{cer} + 1) - q_d/q].\end{aligned}$$

Theorem 4. Assume that six hash functions H_i, for $i \in [1, 6]$, are random oracles and $\mathcal{A}_4$ is a Type IV adversary against the CB-PKEET scheme with advantage ϵ in the security game $G_{\text{CBEET-OW-CCA}}$. Then, there is an algorithms $\mathcal{C}$ to solve the BDH problem with advantage

$$\epsilon' \geq (1/q_{H_3})\,[4(\epsilon - 1/2^{\lambda}) - q_d/q],$$

where q_{H_3} and q_d respectively are query times to random oracle H_3 and decryption queries.

Proof. An algorithm $\mathcal{C}$ is given an instance of the BDH problem: $(\mathcal{G}, U, U^a, U^c, V, V^a, V^b)$ where $\mathcal{G} = (q, G_1, G_2, G_T,$

$\hat{e}$). Let $D = \hat{e}(U, V)^{abc} \in G_T$ be the solution of the BDH problem. The algorithm $\mathcal{C}$ simulates a challenger to find D by interacting with $\mathcal{A}_4$ in the following security game $G_{\text{CBEET-OW-CCA}}$.

- **Setup.** The challenger $\mathcal{C}$ generates the system public parameter $SPP = (\mathcal{G}, U, V, SPK, H_1, H_2, H_3, H_4, H_5, H_6)$ by setting $SPK = U^s$, where $s \in Z_q^*$ is random value as the system secret key SSK. Then, the system public parameter SPP is sent to $\mathcal{A}_4$. Here, $H_1, H_2,..., H_6$ are hash functions as random oracles. Because $\mathcal{C}$'s responses to queries to these random oracles issued from $\mathcal{A}_4$ must be consistent, $\mathcal{C}$ must maintain the several lists, namely L_{H1}, L_{H2}, ..., L_{H6}, L_{Key} which are defined in phase 1 below.
- **Phase 1.** The adversary $\mathcal{A}_4$ may issue the following queries.
  - H_1 *query*. it is similar to the proof of Theorem 1.
  - H_2 *query*. it is similar to the proof of Theorem 3.
  - H_3-H_6 *queries*. it is similar to the proof of Theorem 1.
  - *User public key query*. it is similar to the proof of Theorem 2.
  - *Decryption query*. it is similar to the proof of Theorem 3.
  - *Trapdoor query*. it is similar to the proof of Theorem 3.

Challenge. The adversary $\mathcal{A}_4$ sends a target identity ID^* to $\mathcal{C}$. Then, $\mathcal{C}$ uses ID^* to search the list L_{key} of the form [$ID, USK, UPKA, UPKB, CertA, CertB, cn$]. If cn = 0, $\mathcal{C}$ aborts this game. Otherwise, $\mathcal{C}$ picks a random message $M^* \in \{0, 1\}^\lambda$ and a value $\kappa \in \{0, 1\}^l$, and then uses M^* and κ to issue H_5 *query* to gain θ. In addition, $\mathcal{C}$ uses M^* and $\hat{e}(U^c, V^{a\tau^* x^*})$ to issue H_4 *query* and H_6 *query* to gain γ and η. Further, $\mathcal{C}$ sets $CT_1^* = U^\theta$, $CT_2^* = U^c$, $CT_3^* \in \{0, 1\}^{\lambda+l}$ and $CT_4^* = \gamma^\theta \cdot \eta$. Here, CT_3^* is a random value in $\{0, 1\}^{\lambda+l}$. Finally, $\mathcal{C}$ runs the target ciphertext $CT^* = (CT_1^*, CT_2^*, CT_3^*, CT_4^*)$.

Phase 2. The adversary $\mathcal{A}_4$ can continue to issue the same queries as in phase 1.

Guess. The adversary $\mathcal{A}_4$ sends a message $M' \in \{0, 1\}^\lambda$ as the answer to guess. $\mathcal{A}_4$ wins this game if $M' = rv$. $\mathcal{C}$ randomly selects a tuple [$\mu^*, CT_1^*, CT_2^*, \varphi^*$] from the list L_{H3} of the form [μ, CT_1, CT_2, φ] to gain μ^*. Then, D = $D = (\mu^*)^{(s\hat{}2 x^*\sigma^*)\hat{}-1}$ is outputted as the solution of the BDH problem.

Analysis. We first discuss the simulations of hash functions, namely $H_1, H_2,..., H_6$. Obviously, we can say that the hash functions H_1, H_2, H_4, H_5 and H_6 as the random oracles are perfect simulations since the inputs and outputs of the random oracles are independent of the solution of the BDH problem. Assume that $EventH_3^*$ is a events of issuing the H_3 query with ($\hat{e}(U, V)^{abcs^2x^*\sigma^*}$, CT_1^*, CT_2^*). Here, we say that the hash functions H_3 as the random oracles is perfect simulations if events $EventH_3^*$ did not occur. We then discuss the simulations of the *decryption query*. We denote $EventDecFail$ as the event that the ciphertext is valid, and the challenger $\mathcal{C}$ is unable to decrypt it. The probability of this event is $\Pr[EventDecFail] \le q_d/q$.

Moreover, we denote $Event = (EventH_3^* \vee EventDecFail)|\neg EventAbor$ as the event that this game will not be aborted, where $EventAbort$ is the event that the challenger $\mathcal{C}$ aborts this game. We can obtain probability $\Pr[rv = rv' \,|\neg\, Event] \le 1/2^\lambda$ if the event $Event$ does not occur. Further, we get

$$\begin{aligned}\Pr[rv = rv'] &= \Pr[rv = rv'|Event]\Pr[Event] \\ &\quad + \Pr[rv = rv'|\neg Event]\Pr[\neg Event] \\ &\le \Pr[Event] + (1/2^\lambda)\cdot\Pr[\neg Event] \\ &= \Pr[Event] + (1/2^\lambda)\cdot(1 - \Pr[Event]) \\ &= (1/2^\lambda)\cdot\Pr[Event] + 1/2^\lambda.\end{aligned}$$

According to the sense of ϵ, we have $\epsilon = \Pr[rv = rv'] - 1/2^\lambda$. Hence, we obtain $\epsilon = \Pr[rv = rv'] - 1/2^\lambda \le \Pr[Event] \le (\Pr[EventH_3^*] + \Pr[EventDecFail]) / \Pr[\neg EventAbor]$. By this inequality, we have $\Pr[EventH_3^*] \ge (\epsilon - 1/2^\lambda)\cdot\Pr[\neg EventAbor] - \Pr[EventDecFail]$. Since $\Pr[\neg EventAbor] = \upsilon(1 - \upsilon)$, we can gain $\Pr[\neg EventAbor] \ge 1/4$ when $\upsilon = 1 - 1/(1 + 1)$. We then have $\Pr[EventH_3^*] \ge 4(\epsilon - 1/2^\lambda) - q_d/q$.

If the event $EventH_3^*$ occurs, the adversary A_4 can know the the the target ciphertext CT^* is invalid. $H_3(\hat{e}(U, V)^{abcs^2x^*\sigma^*}, CT_1^*, CT_2^*)$ has been recorded in the list L_{H3}. We can say that the challenger $\mathcal{C}$ wins this game if the correct element was chosen in the list L_{H3}. Therefore, the challenger $\mathcal{C}$ can solve the BDH problem with advantage

$$\begin{aligned}\epsilon' &\ge (1/q_{H_3})\Pr[EventH_3^*] \\ &\ge (1/q_{H_3})\cdot[4(\epsilon - 1/2^\lambda) - q_d/q].\end{aligned}$$

6. Comparison

In this section, we give a comparison of our proposed CBEET scheme with the IBEET scheme [15] and the CBE scheme [19]. We employ two notations to analyze the computational cost of encryption, decryption and equality test. The two notations are defined as below.

- $Cost_{pair}$: the cost of performing a bilinear pairing operation $e: G_1 \times G_2 \to G_T$.
- $Cost_{exp}$: the cost of performing an exponentiation operation in G_1, G_2 or G_T.

We utilize the relevant simulation results [27] to obtain $Cost_{pair} \approx 20$ms and $Cost_{exp} \approx 7$ms. The relevant simulations are executed in a PC environment where the hardware specification is Intel Core i7 CPU 1.80 Ghz processor. The inputs of relevant simulations are a finite field F_q, G_1, G_2 and G_T, where q is a 512-bit prime number, and q is the order for three groups G_1, G_2 and G_T. We also obtain $Cost_{pair} \approx 96$ms and $Cost_{exp} \approx 31$ms in a mobile device environment where the hardware specification is Intel 624-MHz PXA270 CPU.

Table 3 presents the comparisons of our proposed CBEET scheme with the IBEET scheme [15] and the CBE scheme [19] in terms of computational cost, key escrow problem and equality test property. For the computational cost in the procedure of encryption, the CBE scheme [19] is the best, but it does not have the property of providing equality test. On the other hand, the computational cost for decryption and equality test of our CBEET scheme is almost the same as that of the IBEET scheme [15]. However, there exists the key escrow problem in the IBEET scheme. The KGC keeps the private key for each user and may use the private key without the user's knowledge. Conversely, our CBEET scheme not only avoids the key escrow problem, but also retains the performance of encryption, decryption and equality test.

7. Conclusions and Future Work

We introduced a new syntax of CBEET, which incorporates the equality test into CBE. Additionally, novel security notions were proposed. Building upon the syntax of CBEET, we presented the first CBEET scheme. Based on the BDH problem, our scheme has been rigorously proven secure against four distinct adversaries in the $G_{\text{CBEET-IND-CCA}}$ and $G_{\text{CBEET-OW-CCA}}$ security games. This proposed scheme ensures data confidentiality, even when the data is stored on the cloud. The cloud servers are unable to access the content of the data, but still being able to perform tasks of equality test. As compared with the existing IBEET scheme and CBE scheme, our CBEET scheme not only avoids the key escrow problem, but also retains the performance of encryption, decryption and equality test.

When considering future research, it is important to focus on enhancing the proposed scheme's security properties, such as anonymity, mutual authentication, freshness, and resistance to replay attacks. Hence, for future research, it is crucial to address these shortcomings and develop a scheme that ensures mutual authentication, freshness, anonymity, and resistance

Table 3
Comparisons of our CBEET with existing IBEET and CBE

	Ma's IBEET scheme [15]	Shao's CBE scheme [19]	Our CBEET scheme
Computational cost for encryption on a mobile device	$2Cost_{pair} + 6Cost_{exp}$ (≈ 378 ms)	$Cost_{pair} + 2Cost_{exp}$ (≈ 158 ms)	$2Cost_{pair} + 5Cost_{exp}$ (≈ 347 ms)
Computational cost for decryption on a mobile device	$2Cost_{pair} + 2Cost_{exp}$ (≈ 254 ms)	$2Cost_{pair} + 2Cost_{exp}$ (≈ 254 ms)	$2Cost_{pair} + 4Cost_{exp}$ (≈ 316 ms)
Computational cost for equality test on a PC	$4Cost_{pair}$ (≈ 80 ms)	-	$4Cost_{pair}$ (≈ 80 ms)
Avoiding key escrow problem	No	Yes	**Yes**
Possessing equality test property	Yes	No	**Yes**

to replay attacks. This can involve exploring techniques to safeguard user identities, prevent unauthorized access to secret data stored in devices, and implement measures to detect and prevent message tampering during transmission. By addressing these concerns, the proposed scheme can be significantly improved in terms of its overall security properties. Additionally, we should also consider the adversary's capability to steal the user's device and access data from its memory, and even potentially acquire keys for decryption purposes. These security features are worth investigating as part of our research agenda.

Acknowledgement

This research was partially supported by Ministry of Science and Technology, Taiwan, under contract no. MOST 110-2222-E-019-001-MY2 and MOST 110-2221-E-019-041-MY3.

References

1. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D. Key-privacy in Public Key Encryption. In: ASIACRYPT'01, 2001, LNCS 2248, 566-582. https://doi.org/10.1007/3-540-45682-1_33
2. Bellare, M., Waters, B., Yilek, S. Identity-based Encryption Secure Against Selective Opening Attack. In: TCC'11, 2011, LNCS 6597, 235-252. https://doi.org/10.1007/978-3-642-19571-6_15
3. Boneh, D., Crescenzo, G. D., Ostrovsky, R., Persiano, G. Public Key Encryption with Keyword Search. In: EUROCRYPT'04, 2004, LNCS 3027, 506-522. https://doi.org/10.1007/978-3-540-24676-3_30
4. Boneh, D., Franklin, M. Identity-based Encryption from the Weil Pairing. In: CRYPTO'01, 2001, LNCS 2139, 213-229. https://doi.org/10.1007/3-540-44647-8_13
5. Duong, D. H., Fukushima, K., Kiyomoto, S., Roy, P. S., Susilo, W. A Lattice-based Public Key Encryption with Equality Test in Standard Model. In: ACISP'19, 2019, LNSC 11547, 138-155. https://doi.org/10.1007/978-3-030-21548-4_8
6. 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. Theoretical Computer Science, 2022, 905, 31-53. https://doi.org/10.1016/j.tcs.2021.12.013
7. Galindo, D., Morillo, P., Ràfols, C. Improved Certificate-based Encryption in the Standard Model. Journal of systems and software, 2008, 81(7), 1218-1226. https://doi.org/10.1016/j.jss.2007.09.009
8. Gentry, C. Certificate-based Encryption and the Certificate Revocation Problem. In: EUROCRYPT'03, 2003, LNCS 2656, 272-293. https://doi.org/10.1007/3-540-39200-9_17
9. Guo, Y., Li, J., Lu, Y., Zhang, Y., Zhang, F. Provably Secure Certificate-based Encryption with Leakage Resilience. Theoretical Computer Science, 2018, 711, 1-10. https://doi.org/10.1016/j.tcs.2017.10.020
10. Guo, L., Lu, Y., Miao, Q., Zu, G., Wang, Z. An Efficient Certificate-Based Encryption Scheme Without Random Oracles. In: International Conference on Artificial Intelligence and Security'22, 2022, LNCS 13340, 97-107. https://doi.org/10.1007/978-3-031-06791-4_8
11. Huang, K., Tso, R., Chen, Y. C. Somewhat Semantic Secure Public Key Encryption with Filtered-equality-test in the Standard Model and Its Extension to Searchable Encryption. Journal of Computer and System Sciences, 2017, 89, 400-409. https://doi.org/10.1016/j.jcss.2017.06.001
12. Lee, H. T., Ling, S., Seo, J. H., Wang, H., Youn, T. Public Key Encryption with Equality Test in the Standard Model. Information Sciences, 2020, 516, 89-108. https://doi.org/10.1016/j.ins.2019.12.023
13. Li, H., 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. https://doi.org/10.1109/ACCESS.2019.2899680
14. Liao, Y., Chen, H., Huang, W., Mohammed, R., Pan, H., Zhou, S. Insecurity of an IBEET Scheme and an ABEET Scheme. IEEE Access, 2019, 7, 25087-25094. https://doi.org/10.1109/ACCESS.2019.2900752
15. Ma, S. Identity-based Encryption with Outsourced Equality Test in Cloud Computing. Information Sciences, 2016, 328, 389-402. https://doi.org/10.1016/j.ins.2015.08.053
16. Ma, S., Zhang, M., Huang, Q., Yang, B. Public Key Encryption with Delegated Equality Test in a Multi-user Setting. The Computer Journal, 2015, 58(4), 986-1002. https://doi.org/10.1093/comjnl/bxu026
17. Sahai, A., Waters, B. Fuzzy Identity-based Encryption. In: EUROCRYPT'03, 2003, LNCS 3494, 457-473. https://doi.org/10.1007/11426639_27

18. Shamir, A. Identity-based Cryptosystems and Signature Schemes. In: CRYPTO'84, 1984, LNCS 196, 47-53. https://doi.org/10.1007/3-540-39568-7_5

19. Shao, Z. Enhanced Certificate-based Encryption from Pairings. Computers and Electrical Engineering, 2011, 37(2), 136-146. https://doi.org/10.1016/j.compeleceng.2011.01.007

20. Shareef, O. S. F., Sagheer, A. M. Improved Certificate-based Encryption Scheme in the Big Data: Combining AES and (ECDSA-ECDH). Ibn AL- Haitham Journal for Pure and Applied Sciences, 2021, 2021, 82-95. https://doi.org/10.30526/2021.IHICPAS.2655

21. Tang, Q. Public Key Encryption Schemes Supporting Equality Test with Authorisation of Different Granularity. International Journal of Applied Cryptography, 2012, 2(4), 304-321. https://doi.org/10.1504/IJACT.2012.048079

22. Tsai, T. T., Tseng, Y. M., Wu, T. Y. Efficient Revocable Multi-receiver ID-based Encryption. Information Technology and Control, 2013, 42(2), 159-169. https://doi.org/10.5755/j01.itc.42.2.2244

23. Tseng, Y. M., Tsai, T. T., Huang, S. S., Huang, C. P. Identity-based Encryption with Cloud Revocation Authority and Its Applications. IEEE Transactions on Cloud Computing, 2018, 6(4), 1041-1053. https://doi.org/10.1109/TCC.2016.2541138

24. Uyyala, P. Secure Channel Free Certificate-Based Searchable Encryption Withstanding Outside and Inside Keyword Guessing Attacks. The International Journal of Analytical and Experimental Modal Analysis, 2021, 13, 2467-2474.

25. Wu, J. D., Tseng, Y. M., Huang, S. S., Tsai, T. T. Leakage-resilient Certificate-based Key Encapsulation Scheme Resistant to Continual Leakage. IEEE Open Journal of the Computer Society, 2020, 1, 131-144. https://doi.org/10.1109/OJCS.2020.3008961

26. Wu, L., Zhang, Y., Choo, K. K. R., He, D. Efficient and Secure Identity-based Encryption Scheme with Equality Test in Cloud Computing. Future Generation Computer Systems, 2017, 73, 22-31. https://doi.org/10.1016/j.future.2017.03.007

27. Xiong, H., Qin, Z. Revocable and Scalable Certificateless Remote Authentication Protocol with Anonymity for Wireless Body Area Networks. IEEE Transactions on Information Forensics and Security, 2015, 10(7), 1442-1455. https://doi.org/10.1109/TIFS.2015.2414399

28. Yang, G., Tan, C. H., Huang, Q., Wong, D. S. Probabilistic Public Key Encryption with Equality Test. In: CT-RSA'10, 2010, LNCS 5985, 119-131. https://doi.org/10.1007/978-3-642-11925-5_9

29. Yao, J., Li J., Zhang, Y. Certificate-based Encryption Scheme without Pairing. KSII Transactions on Internet and Information Systems, 2013, 7(6), 1480-1491. https://doi.org/10.3837/tiis.2013.06.008

30. Yu, Q., Li, J., Zhang, Y. Leakage-resilient Certificate-based Encryption. Security and Communication Networks, 2015, 8, 3346-3355. https://doi.org/10.1002/sec.1258

31. Yu, Z., Gao, C. Z., Jing, Z., Gupta, B. B., Cai, Q. A Practical Public Key Encryption Scheme Based on Learning Parity with Noise. IEEE Access, 2018, 6, 31918-31923. https://doi.org/10.1109/ACCESS.2018.2840119

32. Zeng, M., Chen, J., Zhang, K., Qian, H. Public Key Encryption with Equality Test via Hash Proof System. Theoretical Computer Science, 2019, 795, 20-35. https://doi.org/10.1016/j.tcs.2019.05.033

33. Zhang, X., Xu, C. Trapdoor Security Lattice-based Public-key Searchable Encryption with a Designated Cloud Server. Wireless Personal Communications, 2018, 100(3), 907-921. https://doi.org/10.1007/s11277-018-5357-6

34. Zhou, Y., Xu, Y., Qiao, Z., Yang, B., Zhang, M. Continuous Leakage-resilient Certificate-based Signcryption Scheme and Application in Cloud Computing. Theoretical Computer Science, 2021, 860, 1-22. https://doi.org/10.1016/j.tcs.2021.01.024

This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 (CC BY 4.0) License (http://creativecommons.org/licenses/by/4.0/).