Public Key Encryption with Equality Test for Heterogeneous Systems in Cloud Computing

Rashad Elhabob[1], Yanan Zhao[1], Iva Sella[1] and Hu Xiong[1,*]

[1] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

[e-mail: rashaduestc@gmail.com]

*Corresponding author: Hu Xiong

Received December 22, 2018; revised March 18, 2019; accepted March 25, 2019; published September 30, 2019

Abstract

Cloud computing provides a broad range of services like operating systems, hardware, software and resources. Availability of these services encourages data owners to outsource their intensive computations and massive data to the cloud. However, considering the untrusted nature of cloud server, it is essential to encrypt the data before outsourcing it to the cloud. Unfortunately, this leads to a challenge when it comes to providing search functionality for encrypted data located in the cloud. To address this challenge, this paper presents a public key encryption with equality test for heterogeneous systems (PKE-ET-HS). The PKE-ET-HS scheme simulates certificateless public encryption with equality test (CLE-ET) with the identity-based encryption with equality test (IBE-ET). This scheme provides the authorized cloud server the right to actuate the equivalence of two messages having their encryptions performed under heterogeneous systems. Basing on the random oracle model, we construct the security of our proposed scheme under the bilinear Diffie-Hellman (BDH) assumption. Eventually, we evaluate the size of storage, computation complexities, and properties with other related works and illustrations indicate good performance from our scheme.

Keywords: Cloud computing, searchable encryption, equality test, heterogeneous systems.

http://doi.org/10.3837/tiis.2019.09.023 **ISSN : 1976-7277**

1. Introduction

Rapid elasticity with high computing services that sustain lowered costs have propagated cloud computing into a sought-after paradigm, due to the standardization, commercialization, and application [1][2][3]. With the massive growth of data, the scope of data storage has augmented. These on-demand attributes have resulted in making available capabilities for the storage of these tremendous amounts of data. Cloud computing offers a virtualized resource pool that uses distributed storage, where the immense data can be accessed with virtual applications over the internet on user demand. However, the knowledge that the cloud server is many times regarded as untrusted raises concerns by users [4][5]. It would be difficult for users to consider storing data that is sensitive to the cloud server. This is because the data accessed by these users is replicated onto specific devices and the requisite to ensure data confidentiality and authentication arises [6][7]. Storing data on a single virtual pool results to difficulty in achieving the same amount of security for this data as compared to the physical network [8]. Hence, the public key infrastructure (PKI) is introduced to enable secure and trusted data sharing on the cloud. The PKI is therefore considered when sensitive data that has to be uploaded to the server, is encrypted using the public key of a receiver and then sent to the cloud server. This ensures the sharing of data is secured, authenticated, and verified in such a way that the authorized user uses his/her secret key to decrypt the secured data. In case encrypted data in massive amounts have been stored in the cloud, the search over these encrypted data is required because it is impractical for the users to download all data from cloud server each time s/he needs the encrypted data. Therefore, public key encryption with search functionality is required to search the encrypted data stored in the cloud server without affecting the privacy of the user.

With this in consideration, to ensure that a user's information is not disclosed whenever their data is searched; search functionality is supported in the ciphertexts that are stored in the cloud server. This allows for the ability to search the ciphertexts, with no information related to the plaintexts being exposed. This idea was first proposed by Boneh *et al*. [9], where the keyword search function was incorporated into public key cryptography and is known as PKE-KS. However, PKE-KS being able to support search functionality still experiences a drawback where the search function only works for ciphertexts encrypted under the same public key.

To handle this drawback, Yang *et al*. [10] presented a scheme known as public key encryption with equality test (PKE-ET). In PKE-ET scheme the equality test can not only be performed on the ciphertexts which are encrypted under the same public key but also under different public keys. Consequently a lot of work has been put into improving the equality test scheme such as [11][12][13][14][15]. However, considering that PKE-ET is founded on public key infrastructure (PKI) system, certificate management becomes an issue since the systems overhead drastically increase. To solve this problem, Ma [16] proposed the notion of identity-based encryption with outsourced equality test (abbreviated as IBE-ET). The IBE-ET scheme is constructed under the IBC cryptosystem. In IBE-ET scheme, the cloud server can perform the equivalence test amid two messages, which have been encrypted under the same identity as well as different identities. Moreover, much effort has been put to improve the idea of IBE-ET such as [17][18][19]. However, IBE-ET schemes still experience difficulties as a result of the key escrow problem. The key escrow problem happens when the user needs to obtain a decryption key, he/she first contacts the private key generator (PKG). The PKG makes use of its master secret key (msk) to generate a decryption key of a user and sends it back to the user.

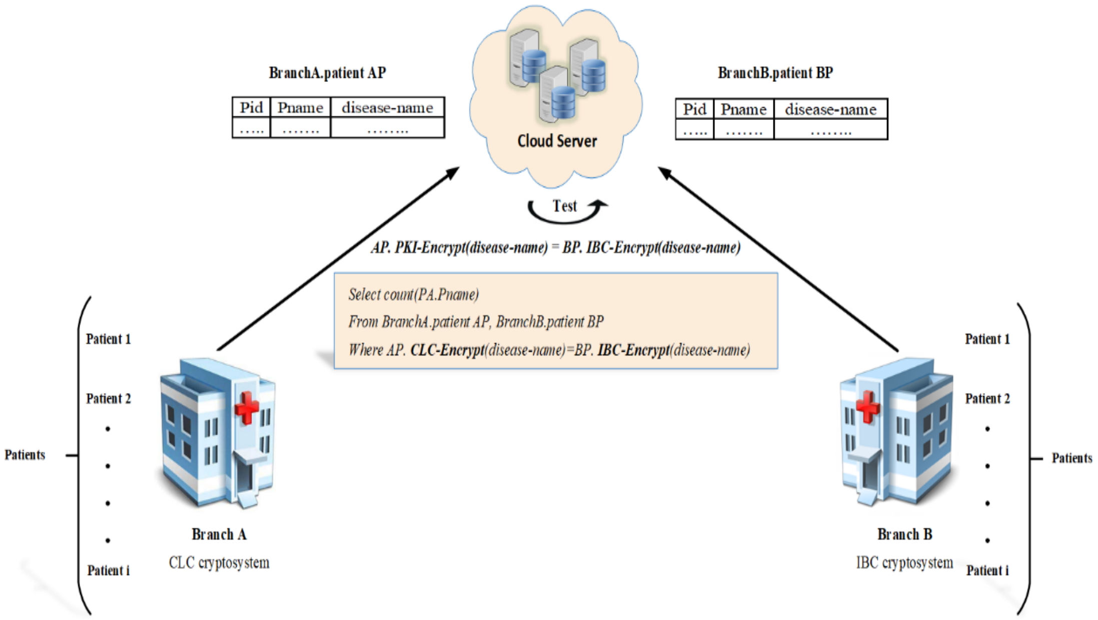

Fig. 1. Application scenario for PKE-ET-HS scheme.

Here the PKG has access to the users' encrypted data because it has their decryption keys. To fix this problem, Al-Riyami *et al*. [20] proposed the notion of certificateless public key encryption (CL-PKE). In this notion, users own decryption keys which are in two parts. The user produces the first part of the key, while the other part is produced by the key generation center (KGC). Therefore, the key generation center (KGC) has partial access to a user's decryption key, ensuring it does not have access to a user's data. By incorporating the idea of PKE-ET and the certificateless public key encryption (CL-PKE), Qu *et al*. [21] proposed the notion of certificateless public key encryption with equality test (CL-PKE-ET). The CL-PKE-ET scheme is constructed under the CLC cryptosystem.

An observation into the above-mentioned schemes indicates their homogeneous nature. Namely, the cloud server can only execute the equality test between two ciphertexts encrypted under the same cryptosystem. Therefore, if we need to delegate the cloud server to perform the equivalence test between ciphertexts encrypted under the different cryptosystems, we should construct a cryptographic scheme that provides a heterogeneous equivalence test. In other words, we should construct a secure scheme that allows the cloud server to perform the equality test between ciphertexts encrypted under the different cryptosystems.

A typical scenario for a heterogeneous systems equality test is shown in **Fig. 1**. In this scenario, we have a hospital that has many branches distributed in different countries. We assume that these countries have different network providers. In the sense that, each country has different security techniques. For example, the branch in the first country uses CLC cryptosystem to protect the security of its network, while the branch in the other country uses IBC cryptosystem. However, all of these branches receive data from their patients, encrypted under the branches' public key and stored in the cloud server of this hospital. Consequently, if the hospital statistics department needs to produce a statistical report to figure out the number of patients in the branch A with the same disease in the branch B, an SQL statement should be written as follows:

```
Select count(PA.Pname)
From BranchA.patient AP, BranchB.patient BP
Where AP.CLC-Encrypt(disease-name) = BP.IBC-Encrypt(disease-name).
```

Considering that the tables of a patient in branch A and branch B are denoted by BranchA.patient and BranchB.patient, respectively, and these two tables contain the same column (e.g., Pid, Pname, disease-name). It is noticeable that, in the "where clause" the cloud server needs to perform the equality test between the ciphertext encrypted under the CLC cryptosystem and the ciphertext encrypted under the IBC cryptosystem. Therefore, branch A and B should send their trapdoors to the cloud server, whereby it performs the equality test and returns the result.

1.1 Our contribution

A novel public key encryption with equality test for heterogeneous systems (PKE-ET-HS) is presented in this paper to deal with the practical needs in the cloud server. In our proposed scheme, we can designate the cloud server to perform equivalence test between the ciphertext encrypted under the CLC cryptosystem with the ciphertext encrypted under the IBC cryptosystem. The contribution made in this paper can be briefly outlined as follows:

1. With the integration between IBE-ET and CLE-ET, the formalized definition and the security model of PKE-ET-HS scheme is presented.

2. Based on the bilinear pairing, our PKE-ET-HS scheme is proposed. In the random oracle model (ROM), the security of our suggested scheme has been proved under the BDH assumption.
3. Finally, in terms of computation and communication costs incurred through storage size, encryption, decryption and testing phases, our scheme is compared with alternative works. The outcome illustrates that our proposed scheme outperforms the existing work.

1. 2 Organization

The other parts of this paper will be as follows; the related works and the preliminaries in Section 2 and 3, respectively. Section 4 will show the definitions. Section 5 will be our presentation of the PKE-ET-HS scheme while the security analysis and performance shall be discussed in section 6 and 7, respectively. We finalize our paper in section 8.

2. Related Work

Boneh *et al*. [9] proposed a way of searching the public key encryption scheme by adopting keywords known as public key encryption with keyword search (PKE-KS). The ciphertexts in this scheme would be run through an equivalence test to determine if the keywords are the same. This is done by a third-party that is considered semi-trusted. Abdalla *et al*. [22] put forth a scheme based on PKE-KS that would adopt the advantages of both IBE and PKE schemes. This scheme was known as identity-based encryption with keyword search (IBE-KS) and supported ciphertexts that were encrypted under the same identity. However, the above-mentioned searchable encryption schemes only support the ciphertexts which are encrypted by the same public key. Yang *et al*. [10] found a solution to this limitation by proposing an encryption scheme that incorporated equality test, not only on ciphertexts encrypted with the same public keys but also with different public keys. This scheme was known as public key encryption with equality test (PKE-ET). The advantage in this scheme is that, it is quite flexible since an authorized cloud server has the search functionality hence can search messages to ascertain whether two ciphertexts encrypted with same or different public keys are equivalent. Consequently, Tang *et al*. [23] found a way which would ensure PKE-ET has authorization enforced which he referred to as fine-grained authorization public key encryption with equality test (FG-PKE-ET). This scheme made provision for only two users to perform equality test. The users, would, however have the assistance of a third-party entity. He further proposed an improvement on the FG-PKE-ET by incorporating two proxy setting [24] into the scheme. The two proxies would cooperate and ensure that the equality test is accomplished. Additionally, Tang proposed a scheme known as all-or-nothing PKE-ET [11]. This scheme was a more refined one and it could choose who would have the right to perform equality test on a coarser granularity manner. Nevertheless, there were situations where delegated parties were the only ones required to finish work in practical multi-user settings. Ma *et al*. [12] introduced the notion of PKE incorporating delegated equality test (PKE-DET). Subsequently, Huang *et al*. [13] proposed a scheme that involved authorized equality test i.e. PKE-AET. Here the given users have the chance of testing equivalence between two ciphertexts or rather two specified ciphertexts. Ma *et al*. [14] improved the PKE-AET by proposing a scheme that supported flexible authorization known as PKE-ET-FA. The urge to keep improving these schemes by researches resulted in the introduction of the PKE-ET into the 5G networks field by Xu *et al*. [15]. This scheme provides users with providence to check if a specified cloud server has correctly performed the equality test on the given ciphertexts. A closer investigation on the above-mentioned schemes, we realize that all of these schemes have their basis on the public key infrastructure (PKI). Unfortunately, the PKI has proved to be unreliable when it comes to scalability since the distribution of public keys is unmanageable. New research areas on identity-based encryption have hence sprouted as shown by [16][17][19]. These encryption schemes have considered the outsourcing of equality tests in order to make the process more flexible when it comes to certificates management. And despite this encryption suffering from key escrow, Qu *et al*. [21] proposed the certificateless public key encryption with equality test (CL-PKE-ET) scheme by integrating the notion of CL-PKE with PKE-ET. As of now we recognize, the equality test for heterogeneous systems has not been brought to literature yet.

3. Preliminaries

We depict the basic definition and properties of the bilinear pairings and Bilinear Diffie-Hellman (BDH) assumption in this section.

3.1 Bilinear map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Suppose that g is a generator of $\mathbb{G}_1$. A bilinear map $e:\mathbb{G}_1\times\mathbb{G}_1\rightarrow\mathbb{G}_2$ satisfies the following properties:

1. Bilinearity: For all $g,h\in\mathbb{G}_1$, for all $x,y\in\mathbb{Z}_p^*$, $e(g^x,h^y)=e(g,h)^{xy}$.
2. Non-degeneracy: For all $g,h\in\mathbb{G}_1$, $e(g,h)\neq 1_{\mathbb{G}1}$.
3. Computability: An efficient algorithm to compute $e(g,h)$ is realizable.

3. 2 Bilinear Diffie-Hellman (BDH) assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups with a large prime order p. Let $e:\mathbb{G}_1\times\mathbb{G}_1\rightarrow\mathbb{G}_2$ be an admissible bilinear map and let g be a generator of $\mathbb{G}_1$. The BDH problem in $<p,\mathbb{G}_1,\mathbb{G}_2,e>$ is as follows: Given $<g,g^a,g^b,g^c>$ for random $a,b,c\in\mathbb{Z}_p^*$, any randomized algorithm $\mathcal{A}$ computes $S=e(g,g)^{abc}\in\mathbb{G}_2$ with an advantage:

$$\mathbf{Adv}_{\mathcal{A}}^{BDH} = Pr[\mathcal{A}(g, g^a, g^b, g^c) = S].$$

We say that the BDH assumption holds in $< p, \mathbb{G}_1, \mathbb{G}_2, e >$ if for any polynomial-time algorithm $\mathcal{A}$, its advantage $\mathbf{Adv}_{\mathcal{A}}^{BDH}$ is negligible.

4. Definitions

The system model and the security model of public key encryption with equality test for heterogeneous systems (PKE-ET-HS) scheme are presented in this section.

4. 1 System model

Fig. 2 illustrates the system model of PKE-ET-HS. The PKE-ET-HS model is made up of four entities: the cloud server, the key generation center (KGC), user A and user B. User A belongs to the CLC cryptosystem while User B to the IBC cryptosystem. Under the CLC cryptosystem, User A sends his/her identity to the KGC, and the KGC generates a corresponding partial secret key (D) and delivers it back to User A. Thereupon, User B found in the IBC cryptosystem sends his/her identity to the KGC which in return sends back a corresponding secret key. Both User A and B then use their secret keys to compute their individual trapdoors denoted by td_{CLC} and td_{IBC} respectively. User A then uses his/her own public key to encrypt data and then outsources the data together with the trapdoor to the cloud server for storage. Consequently, User B uses his/her own ID to encrypt the data and then outsource it together with the trapdoor to cloud server. Since the cloud server has now acquired both td_{CLC} and td_{IBC}, it can now perform an equivalence test between the CLC cryptosystem and IBC cryptosystem.

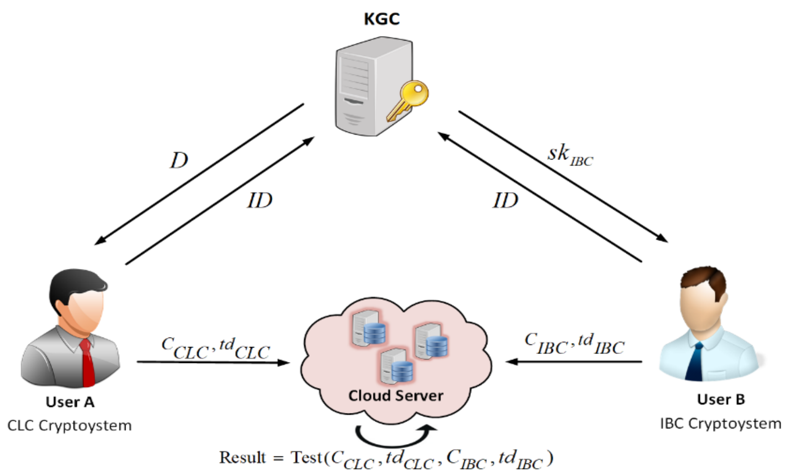

Fig. 2. System Model of PKE-ET-HS.

4. 2 Definition of PKE-ET-HS

A heterogeneous CLC and IBC equality test scheme is made up of the following algorithms.

1. **Setup:** The algorithm uses as input a security parameter λ, and outputs the system parameters, containing the public parameters *PubP* and a master secret key *msk*.
2. **CLC-PKG:** To prompt the secret key of CLC cryptosystem, this algorithm functions as follows:
   - *Generate partial secret key* : The key generation center (**KGC**) runs this algorithm. It uses *PubP*, *msk*, and a public identity of a user $ID \in \{0,1\}^*$ as input, and returns a partial private key D.
   - *Assign secret value:* The user runs this algorithm. It uses *PubP* and **ID** as inputs, and conveys the user's secret value x.
   - *Assign secret key:* The user runs this algorithm. It uses PubP, D, and x as inputs, and returns the user's secret key sk_{CLC}.
   - *Assign public key:* The user runs this algorithm. It uses as inputs *PubP* and sk and returns the user's public key pk.
3. **CLC-Trapdoor:** This is a trapdoor algorithm for CLC users. It uses as input sk_{CLC} of the user in CLC cryptosystem, and returns a trapdoor td_{CLC}.
4. **IBC-PKG:** This is an algorithm that generates the private key for IBC users. The user sends an identity **ID** to its PKG where it computes a corresponding secret key sk_{IBC} and sends it to the user.
5. **IBC-Trapdoor:** This is a trapdoor algorithm for IBC users. It takes as input sk_{IBC} of the user in IBC cryptosystem, and returns a trapdoor td_{IBC}.
6. **CLC-Encrypt:** The CLC users run this algorithm. It utilizes the *PubP*, a message *M*, and public key pk_{CLC} as inputs. The algorithm returns a ciphertext C_{CLC}.
7. **CLC-Decrypt:** The CLC users run this algorithm. It utilizes a ciphertext *C* and a user's secret key sk_{CLC} as inputs, and outputs the plaintext *M*.
8. **IBC-Encrypt:** The IBC users run this algorithm. It takes as inputs a message *M* and an identity **ID**, then it outputs a ciphertext C_{IBC}.
9. **IBC-Decrypt:** The IBC users run this algorithm. It takes as inputs a ciphertext *C* and a secret key sk_{IBC}, then it outputs the plaintext *M*.
10. **Test:** The cloud server runs this algorithm. It uses as inputs a ciphertext C_{IBC} and a trapdoor td_{IBC} for the user in the IBC cryptosystem. Furthermore, it uses as inputs a ciphertext C_{CLC} and a trapdoor td_{CLC} for the user in the CLC cryptosystem. Then, the **Test** algorithm returns 1 if C_{IBC} and C_{CLC} consist of the same message. Otherwise, it returns 0.

4. 3 Security models

According to [16], one-way chosen-ciphertext attack (OW-CCA) security against the adversary in PKE-ET-HS is defined. For simplicity, **PKE-ET-HS-CLC** to denote the situation that users belong to the CLC cryptosystem, and **PKE-ET-HS-IBC** to denote the situation that users belong to the IBC cryptosystem are used.

Definition 1. *For the security of **PKE-ET-HS-CLC**, we consider adversaries of two kinds. Type-1 adversary* $\mathcal{A}_1$ *cannot retrieve the system's master secret key, but has the ability to replace any user's public key. Type-2 adversary* $\mathcal{A}_2$ *has no ability to replace a user's public key, but can retrieve the system's master secret key. **PKE-ET-HS-CLC**'s security model is expounded by the following two games:*

Game 1: Given a security parameter λ, The game between $\mathcal{A}_1$ and the challenger is illustrated as follows:

1. **Setup:** The challenger creates the public parameters *PubP* and the master secret key *msk*. Finally, the challenger returns *PubP*.
2. **Phase 1:** The $\mathcal{A}_1$ is permitted to issue the following queries:
   - *Partial secret key queries* $\langle ID_i \rangle$: The challenger sends D_i to $\mathcal{A}_1$.
   - *Secret key queries* $\langle ID_i \rangle$: The challenger sends sk_{CLC_i} to $\mathcal{A}_1$.
   - *Public key queries* $\langle ID_i \rangle$: The challenger sends pk_{CLC_i} to $\mathcal{A}_1$.
   - *Replace public key queries* $\langle ID_i, pk'_{CLC_i} \rangle$: The challenger replaces the public key pk_{CLC} of the corresponding user with pk'_{CLC_i}.
   - *Decryption queries* $\langle ID_i, C_i \rangle$: This algorithm is run by the challenger in **CLC-Decrypt**(C_i, sk_{CLC_i}), where sk_{CLC_i} is the secret key corresponding to ID_i. Finally, the challenger gives M_i to $\mathcal{A}_1$.
   - *Trapdoor queries:* The challenger creates the trapdoors td_{CLC_i} and td_{IBC_i} by using **CLC-Trapdoor** and **IBC-Trapdoor** algorithms, respectively. Finally, the challenger gives td_{CLC_i} and td_{IBC_i} to $\mathcal{A}_1$.
3. **Challenge:** The challenger then chooses the plaintext $M \in \mathbb{G}_1^*$ randomly and computes $C' = \textbf{CLC-Encrypt}\ (ID_{ch}, M)$. Finally, the challenger sends C' to $\mathcal{A}_1$ as its challenge ciphertext.
4. **Phase 2:** The challenger's response to $\mathcal{A}_1$ is similar to that in **Phase 1** on the grounds that:
   - ID_{ch} is not queried in the *Secret key queries*.
   - If the public key associated with ID_{ch} is replaced, the ID_{ch} should not be queried in the *Partial secret key queries*.
   - If the public key of the user is replaced, the corresponding identity ID_i should not be queried in the *Secret key queries*.
   - (ID_{ch}, C') is not queried in the *Decryption queries*.
5. **Guess:** $\mathcal{A}_1$ outputs M', and wins if $M' = M$. The advantage of $\mathcal{A}_1$ in the game above is defined as follows:

$$\mathbf{Adv}_{PKE-ET-HS,\mathcal{A}_1}^{OW-CCA,\textbf{PKE-ET-HS-CLC}}(\lambda) = Pr[M = M'].$$

Game 2: Provided with a security parameter λ, The game between $\mathcal{A}_2$ and the challenger is expounded as follows:

1. **Setup:** The challenger creates the public parameters *PubP* and the master secret key *msk*. Finally, the challenger gives *PubP* and the *msk* to $\mathcal{A}_2$.
2. **Phase 1:** $\mathcal{A}_2$ issues queries as in **Game 1**, except the *Partial secret key queries* and the *Replace public key queries* should not be issued in this game.
3. **Challenge:** The challenger randomly picks the plaintext $M \in \mathbb{G}_1^*$ and computes $C' = \textbf{CLC-Encrypt}(ID_{ch}, M)$. Finally, the challenger gives C' to $\mathcal{A}_2$ as its challenge ciphertext.
4. **Phase 2:** The challenger's response to $\mathcal{A}_2$ is similar to that in **Phase 1** on grounds that:
   - ID_{ch} is not queried in the *Secret key queries*.
   - (ID_{ch}, C^*) is not queried in the *Decryption queries*.
5. **Guess:** $\mathcal{A}_2$ outputs M', and wins if $M' = M$. Therefore, the advantage $\mathcal{A}_2$ has in the game is:

$$\mathbf{Adv}_{PKE-ET-HS,\mathcal{A}_2}^{OW-CCA,\textbf{PKE-ET-HS-CLC}}(\lambda) = Pr[M = M'].$$

Definition 2. *A **PKE-ET-HS-IBC** scheme possesses the OW-CCA property if no polynomial bounded adversary* $\mathcal{A}$ *has a non-negligible advantage in the following game.*

1. **Setup:** The challenger takes as input a security parameter λ, and executes the **Setup** algorithm. Then, challenger delivers the system parameters to $\mathcal{A}$ and keeps the *msk* secret.
2. **Phase 1:** $\mathcal{A}$ has the permission to administer the following queries.
   - *Key generation query* $\langle ID_i \rangle$: The challenger runs **IBC-PKG** and sends sk_{IBC} to $\mathcal{A}$.
   - *Decryption query* $\langle ID_i, C_i \rangle$: The challenger runs **IBC-Decrypt**(C_i, sk_{IBC_i}) algorithm and sends the result *M* to $\mathcal{A}$.
   - *Trapdoor query* $\langle ID_i \rangle$: The challenger generates the trapdoors td_{IBC_i} and td_{CLC_i} by using **IBC-Trapdoor** and **CLC-Trapdoor** algorithms, respectively. Finally, the challenger sends td_{IBC_i} and td_{CLC_i} to $\mathcal{A}$.
3. **Challenge:** When $\mathcal{A}$ decides the **Phase 1** is finished. A challenger randomly chooses a plaintext $M \in \mathbb{G}_1$, the challenger then sets $C^* = \textbf{IBC-Encrypt}\ (ID^*, M)$. Furthermore, the challenger generates a trapdoor td_{IBC} associated with sk_{IBC} by

using **IBC-Trapdoor** algorithm. Finally, the challenger sends (C^*, td_{IBC}) to $\mathcal{A}$.

4. **Phase 2:** In this phase, the response of the challenger to $\mathcal{A}$ is similar of that one obtained in **Phase 1**. The following constraints are considered.
   - $\langle ID^* \rangle$ is not queried in the *key generation query*.
   - $\langle ID^*, C^* \rangle$ is not queried in the *Decryption query*.

5. **Guess:** $\mathcal{A}$ outputs $M' \in \mathbb{G}_1^*$, and wins if $M = M'$. The advantage that $\mathcal{A}$ has in the game above is defined as follows:

$$\mathbf{Adv}_{PKE-ET-HS,\mathcal{A}}^{OW-CCA,\textbf{PKE-ET-HS-IBC}}(\lambda) = Pr[M = M'].$$

5. Construction

The concrete constructions of heterogeneous systems public key encryption with equality test is instituted in this section.

1. **Setup:** Provided a security parameter λ, the algorithm runs as follows:
   - Generate the pairing parameters: two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order p, and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then choose a random generator $g \in \mathbb{G}_1$.
   - Determine cryptographic hash functions: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \mathbb{G}_1$, $H_3 : \mathbb{G}_2 \rightarrow \{0,1\}^{n_1+n_2}$, where $n_1 = |\mathbb{G}_1|$ and $n_2 = |\mathbb{Z}_p^*|$.
   - Randomly choose $(s_1, s_2) \in \mathbb{Z}_p^*$, then set $g_1 = g^{s_1}$ and $g_2 = g^{s_2}$. The **CLC-PKG** publishes system parameters $\langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2, H_3 \rangle$ and keeps the master secret key (s_1, s_2) secret.
2. **CLC-PKG:** This algorithm generates public and secret key, and functions as follows:
   - *Generate partial secret key*: Given a string ID $\in \{0,1\}^*$:
     - o Compute $h_{ID} = H_1(ID) \in \mathbb{G}_1$.
     - o Compute partial secret key $D = (D_1, D_2) = (h_{ID}^{s_1}, h_{ID}^{s_2})$, where (s_1, s_2) is the master secret key.
   - *Assign secret value:* The algorithm uses as inputs *PubP* and *D*. It chooses $x \in \mathbb{Z}_p^*$ randomly then returns x as a secret value.
   - *Assign secret key:* The algorithm uses as inputs *PubP*, *D*, and x. It computes $sk_{CLC} = (sk_1, sk_2) = (D_1^x, D_2^x)$.

- *Assign public key:* The algorithm uses as inputs *PubP* and a secret value x. It returns public key $pk_{CLC} = (X, pk_1, pk_2) = (g^x, g_1^x, g_2^x)$.

3. **CLC-Trapdoor:** This algorithm takes as input sk of a user in the CLC cryptosystem and outputs a trapdoor $td_{CLC} = sk_1 = D_1^x$.
4. **IBC-PKG:** A user in the IBC cryptosystem sends its identity ID to its **IBC-PKG**. The **IBC-PKG** computes $h_{ID} = H_1(ID)$ and then computes a secret key $sk_{IBC} = (sk_1, sk_2) = (h_{ID}^{s_1}, h_{ID}^{s_2})$.
5. **IBC-Trapdoor:** It takes as input sk_{IBC} of a user in the IBC cryptosystem and outputs a trapdoor $td_{IBC} = sk_1 = h_{ID}^{s_1}$.
6. **CLC-Encrypt:** This algorithm proceeds as follows:
   - Take the message $M \in \mathbb{G}_1^*$, the identity ID, and the public key $pk = (X, pk_1, pk_2)$ as inputs.
   - Verify that if $pk = (X, pk_1, pk_2) \in \mathbb{G}_1^*$, $e(X, g_1) = e(pk_1, g)$, and $e(X, g_2) = e(pk_2, g)$. If these verifications pass, perform the encryption. Otherwise, terminate the encryption.
   - Compute $h_{ID} = H_1(ID) \in \mathbb{G}_1^*$.
   - Pick two random numbers $(r_1, r_2) \in \mathbb{Z}_p^*$.
   - Compute $C = (C_1, C_2, C_3, C_4)$, where $C_1 = g^{r_1}$, $C_2 = g^{r_2}$, $C_3 = M^{r_1} \cdot H_2(e(h_{ID}, pk_1)^{r_1})$, and $C_4 = (M||r_1) \oplus H_3(e(h_{ID}, pk_2)^{r_2})$
7. **CLC-Decrypt:** The algorithm uses as inputs a ciphertext C and a secret key sk and conveys the plaintext M by functioning as follows:
   - Compute $C_4 \oplus H_3(e(sk_2, C_2)) = C_4 \oplus H_3(e(D_2^{x_{ID}}, g^{r_2})) = C_4 \oplus H_3(e(h_{ID}, g_2^{x_{ID}})^{r_2}) = (M||r_1)$.
   - Verify if $C_1 = g^{r_1}$ and $\frac{C_3}{M^{r_1}} = H_2(e(sk_1, C_1))$.
   - If both verifications pass, return M. Otherwise, return the symbol $\perp$.
8. **IBC-Encrypt:** This algorithm uses as inputs a message $M \in \mathbb{G}_1$ and the identity ID of a user in the IBC cryptosystem. It then selects two random numbers $(r_1, r_2) \in \mathbb{Z}_p^*$ and computes $C = (C_1, C_2, C_3, C_4)$, where $C_1 = g^{r_1}$, $C_2 = g^{r_2}$, $C_3 = M^{r_1} \cdot H_2(e(h_{ID}, g_1)^{r_1})$, and $C_4 = (M||r_1) \oplus H_3(e(h_{ID}, g_2)^{r_2})$.
9. **IBC-Decrypt:** This algorithm uses as inputs a ciphertext C and a secret key sk_{CLC} of a user in the IBC cryptosystem. Then, it returns the plaintext *M* by computing $M||r_1 \leftarrow C_4 \oplus H_3(e(dk_2, C_2))$, and then verifies both $C_1 = g^{r_1}$ and $\frac{C_3}{M^{r_1}} = H_2(e(sk_1, C_1))$. If both verifications pass, it returns *M*. Otherwise, it returns the symbol $\perp$.

10. Test $(C_i, td_{CLC}, C_j, td_{IBC})$: Let U_i and U_j be two users of a heterogeneous systems. Let U_i be a user in the CLC cryptosystem and U_j be a user in the IBC cryptosystem. Let $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$ and $C_j = (C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4})$ be the ciphertexts of U_i and U_j, respectively. The **Test** algorithm for heterogeneous systems works as follows:

$$\begin{aligned} Q_i &= \frac{C_{i,3}}{H_2(e(td_{CLC}\ ,\ C_{i,1}))} \\ &= \frac{M_i^{r_{i,1}} \cdot H_2(e(h_{ID,i}\ ,\ pk_{i,1})^{r_{i,1}})}{H_2(e(sk_{i,1}\ ,\ C_{i,1}))} \\ &= \frac{M_i^{r_{i,1}} \cdot H_2(e(h_{ID,i}\ ,\ g_1^{x_i})^{r_{i,1}})}{H_2(e(D_{i,1}^{x_i}\ ,\ g^{r_{i,1}}))} \\ &= \frac{M_i^{r_{i,1}} \cdot H_2(e(h_{ID,i}\ ,\ g_1^{x_i})^{r_{i,1}})}{H_2(e(h_{ID,i}\ ,\ g_1^{x_i})^{r_{i,1}})} \\ &= M_i^{r_{i,1}} \end{aligned}$$

$$\begin{aligned} Q_j &= \frac{C_{j,3}}{H_2(e(td_{IBC}, C_{j,1}))} \\ &= \frac{M_j^{r_{j,1}}.H_2(e(h_{ID,j}, g_1)^{r_{j,1}})}{H_2(e(sk_{j,1}, C_{j,1}))} \\ &= \frac{M_j^{r_{j,1}}.H_2(e(h_{ID,j}, g_1)^{r_{j,1}})}{H_2(e(h_{ID,j}, g_1)^{r_{j,1}})} \\ &= M_j^{r_{j,1}} \end{aligned}$$

The **Test** algorithm returns 1 if $e(C_{i,1}, Q_j) = e(C_{j,1}, Q_i)$. Otherwise, it returns the symbol $\perp$.

6. Security analysis

In this section, the security of the **PKE-ET-HS** scheme is presented. The basic notion of security proof is alike the scheme in [20] and [25].

6. 1 PKE-ET-HS-CLC

Theorem 6.1.1: Presuming that H_1, H_2, H_3 are random oracles and assume that the BDH problem is hard. Therefore, our **PKE-ET-HS-CLC** is OW-CCA secure. Significantly, assume there is a **Type-1 adversary** $\mathcal{A}_1$ that has advantages $\epsilon_1(\lambda)$ against the **PKE-ET-HS-CLC**. Assume that $\mathcal{A}_1$ makes q_{pk} public key queries, q_{sk} secret key queries, q_{psk} partial secret key queries, q_t trapdoor queries, q_{rpk} replace public key queries, q_{dec} decryption queries, q_{H_2} hash queries to H_2, and q_{H_3} hash queries to H_3. Thus, we have an algorithm $\mathcal{B}_1$ which breaks BDH problem with advantage at least $\epsilon_1(\lambda)/e(q_{sk}+q_{psk}+q_t+1).q_{H_3}$.

Proof: Presimung that $\mathcal{B}_1$ is given as inputs the BDH parameters $\langle p, \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and arbitrary instance $< g, g^a, g^b, g^c >$ of the BDH problem, where g is a random generator of $\mathbb{G}_1$ and $a, b, c \in_R \mathbb{Z}_p^*$. Suppose that $e(g,g)^{abc} \in \mathbb{G}_2$ is the solution of BDH problem. Then, we demonstrate how an algorithm $\mathcal{B}_1$ obtains $e(g,g)^{abc}$ by reacting with $\mathcal{A}_1$ as follows:

1. **Setup:** The algorithm $\mathcal{B}_1$ gives $\mathcal{A}_1$ the public parameters $PubP = \langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2, H_3\rangle,$ where $g_1 = g_2 = g^a$.
2. **Phase 1:** At any time $\mathcal{A}_1$ can make queries to $q_{pk}, q_{sk}, q_{psk}, q_t$, $q_{rpk}, q_{dec}, q_{H_2}$, or q_{H_3}. To respond to these queries $\mathcal{B}_1$ works as follows:
    - H_1**-queries**(ID_i)**:** $\mathcal{B}_1$ creates a list of tuples $< ID_i, d_i, z_i, c_i >$ denoted by H_1^{list}. Here, $c_i \in \{0,1\}$, where 1 represents the probability of δ and 0 represents the probability of $1-\delta$. If $c_i = 0$, $\mathcal{B}_1$ returns $d_i = (g_1^{z_i}, g_2^{z_i})$. Otherwise, if $c_i = 1$, $\mathcal{B}_1$ returns $d_i = (g_1^{bz_i}, g_2^{bz_i})$.
    - H_2**-queries**(w_i)**:** $\mathcal{B}_1$creates a list of tuples $< w_i, l_i >$ denoted by H_2^{list}. If w_i is already stored in H_2^{list}, $ \mathcal{B}_1 $ responds with $H_2(w_i) = l_i$. Otherwise, $\mathcal{B}_1$ chooses $l_i \in_R \mathbb{G}_1$ and records item $[w_i, l_i]$ in H_2^{list}. Then, $\mathcal{B}_1$ returns l_i to $\mathcal{A}_1$.
    - H_3**-queries**(v_i)**:** $\mathcal{B}_1$ creates a list of tuples $< v_i, h_i >$ denoted by H_3^{list}. If v_i is already stored in H_3^{list}, $\mathcal{B}_1$ responds with $H_3(v_i) = h_i$. Otherwise, $\mathcal{B}_1$ chooses $h_i \in_R \{0,\}^{n_1+n_2}$ and records item $[v_i, h_i]$ in H_3^{list}. Then, $\mathcal{B}_1$ returns h_i to $\mathcal{A}_1$.
    - **Public key queries**(ID_i): Algorithm $\mathcal{B}_1$ prepares a list of tuples $< ID_i, x_i, D_i, pk_i, sk_i, c_i >$ denoted by PSK^{list} and responds as follows:
        - Check the H_1^{list}, if $c_i = 0$, $\mathcal{B}_1$ executes the *Assign Secret Value* algorithm to compute x_i, then calculates $D_i = (D_{i,1}, D_{i,2})$ $= (g_1^{z_i}, g_2^{z_i})$, $sk_i = (sk_{i,1}, sk_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and $pk_i = (X_i, pk_{i,1}, pk_{i,2}) = (g^{x_i}, g_1^{x_i}, g_2^{x_i})$ by executing *Extract partial secret key*, *Assign secret key*, and *Assign public key algorithms*, respectively. Finally, $\mathcal{B}_1$ records item $[ID_i, x_i, D_i, pk_i, sk_i, 0]$into PSK^{list} and responds to $\mathcal{A}_1$ with pk_i.
        - Otherwise, if $c_i = 1$, $\mathcal{B}_1$ executes the *Assign Secret Value* algorithm to compute x_i , calculates

$pk_i = (X_i, pk_{i,1}, pk_{i,2}) = (g^{x_i}, g_1^{x_i}, g_2^{x_i})$ by executing *Assign public key* algorithm, and records item $[ID_i, x_i, *, pk_i, *, 1]$ into PSK^{list} and responds to $\mathcal{A}_1$ with pk_i.

- **Partial secret key queries**(ID_i)**:** If $c_i = 0$, $\mathcal{B}_1$ returns D_i associated with ID_i from PSK^{list}. Otherwise, if $c_i = 1$, $\mathcal{B}_1$ aborts and fails.
- **Secret key queries**(ID_i)**:** If $c_i = 0$, $\mathcal{B}_1$ returns sk_i associated with ID_i from PSK^{list}. Otherwise, if $c_i = 1$, $\mathcal{B}_1$ aborts and fails.
- **replace public key queries**(ID_i, pk'_i)**:** Suppose that $pk'_i = (X'_i, pk'_{i,1}, pk'_{i,2})$. Algorithm $\mathcal{B}_1$ verifies if $pk'_i = (X'_i, pk'_{i,1}, pk'_{i,2}) \in \mathbb{G}_1^*$ and $e(X'_i, g_1) = e(pk'_{i,1}, g)$ and $e(X', g_2) = e(pk'_{i,2}, g)$. If these verifications pass, $\mathcal{B}_1$ replaces pk_i with pk'_iOtherwise, $\mathcal{B}_1$ returns the symbol $\perp$ to $\mathcal{A}_1$.
- **Decryption queries**(ID_i, C_i)**:** Let $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$. $\mathcal{B}_1$ searches on the PSK^{list} and reacts as follows:
  - If $c_i = 0$ and public key is not replaced, $\mathcal{B}_1$ executes **CLC-Decrypt**(C_i, sk_i) and returns M_i to $\mathcal{A}_1$.
  - Otherwise, for each item in H_3^{list} $\mathcal{B}_1$ responds as follows:
    - Calculate $M_i || r_{i,1} = C_{i,4} \oplus h_i$.
    - Verify if $C_{i,1} = g^{r_{i,1}}$ and $\frac{C_{i,3}}{M_i^{r_{i,1}}} = H_2(e(sk_{i,1}, C_{i,1}))$.
    - If both verifications pass, returnM_i. Otherwise, return the symbol $\perp$.
- **Trapdoor queries**: The Algorithm $\mathcal{B}_1$'s response to the queries is illustrated as follows:
  - In case of **PKE-ET-HS-CLC**, $\mathcal{B}_1$ searches on PSK^{list} with respect to the ID_i to get $sk_{CLC_i} = (g_1^{z_i x_i}, g_2^{z_i x_i})$ and responds to $\mathcal{A}_1$ with $td_{CLC_i} = g_1^{z_i x_i}$.
  - In case of **PKE-ET-HS-IBC**, $\mathcal{B}_1$ uses sk_{IBC_i} to run **IBC-Trapdoor** algorithm, and then sends $td_{IBC_i} = td_{IBC}$ to $\mathcal{A}_1$.

3. **Challenge:** When $\mathcal{A}_1$ decides **Phase 1** is finished. It outputs identity ID_{ch} on which it intends to be challenged on. The algorithm $\mathcal{B}_1$ chooses $M' \in_R \mathbb{G}_1^*$, then searches on PSK^{list} and reacts as follows:
   - If $C_i = 0$, terminate with failure.
   - Else, the following steps are performed:
     - Choose $C_4 \in_R \{0,1\}^{n_1+n_2}$ and set $C_2 = g^c$.
     - Generate $C' = (C_1, C_2, C_3, C_4)$ and send to $\mathcal{A}_1$ as a challenge ciphertext.
   - The decryption of C' will then be:

$$\begin{aligned} M^*||r &= C_4 \oplus H_3(e(sk_{i,2}, C_2)) \\ &= C_4 \oplus H_3(e(g_2^{bz_ix_i}, g^c)) \\ &= C_4 \oplus H_3(e(g^{abz_ix_i}, g^c)) \\ &= C_4 \oplus H_3(e(g,g)^{abcz_ix_i}). \end{aligned}$$

4. **Phase 2:** In this phase, the response of the challenger to $\mathcal{A}_1$ is similar to that one obtained in **Phase 1**. The following grounds are considered.
    - ID_{ch} is not queried in the **Secret key queries**.
    - If the public key corresponding to ID_{ch} is replaced, the ID_{ch} should not be queried in the **Partial secret key queries**.
    - If the public key of the user is replaced, the corresponding identity ID_i is not queried in the **Secret key queries**.
    - (ID_{ch}, C') should not be queried in the **Decryption queries**.
5. **Guess:** $\mathcal{A}_1$ returns M' for M^*. $\mathcal{B}_1$ picks item $[v_i, h_i] \in_R H_3^{list}$ and returns $h_i^{(zx)^{-1}} = e(g,g)^{abc}$ as the solution of BDH problem.

Claim: If $\mathcal{B}_1$ holds in the simulation, then $\mathcal{A}_1$ is viewed similar to real attack. Thus, the $Pr[M^* = M'] \geq \epsilon_1$.

Proof: H_1**-queries** responds as in the real attack. All replies to secret key queries q_{sk}, partial secret key queries q_{psk}, trapdoor queries q_t are valid. Then, the probability of $\mathcal{A}_1$ is $Pr[M = M^{'}] \geq \epsilon_1$. Then, we compute the probability of $\mathcal{B}_1$'s failure in the simulation as follows:

1. The probability that $\mathcal{B}_1$ holds in **Phase 1** or **Phase 2** is equal to $\delta^{q_{sk}+q_{psk}+q_t}$.
2. The probability that $\mathcal{B}_1$ holds in **Challenge** phase is equal to $1-\delta$.
3. The combination of both gives the probability of $\mathcal{B}_1$ holds in the simulation is equal to $\delta^{q_{sk}+q_{psk}+q_t}(1-\delta)$.
4. The maximum of this probability is equal to $\delta_{opt} = 1 - 1/(q_{sk} + q_{psk} + q_t + 1)$.
5. By using δ_{opt}, the probability that $\mathcal{B}_1$ holds is at least $1/e(q_{sk} + q_{psk} + q_t + 1)$.

According to above analysis, the advantage of $\mathcal{B}_1$ is at least:

$$\epsilon_1(\lambda)/e(q_{sk} + q_{psk} + q_t + 1) \quad (1)$$

In addition, the algorithm $\mathcal{B}_1$ is emulating the real attack environment to $\mathcal{A}_1$. Thus, $\mathcal{B}_1$ returns $e(g,g)^{abc}$ with probability at least:

$$\epsilon_1(\lambda)/q_{H_3}. \quad (2)$$

By combining Equations (1) and (2), we have

$$\epsilon_1(\lambda)/e(q_{sk} + q_{psk} + q_t + 1).q_{H_3}.$$

The proof of **Theorem 6.1.1** is completed.

Theorem 6.1.2: Presuming that H_1, H_2, H_3 are random oracles and assume that the BDH problem is hard. Therefore, our **PKE-ET-HS-CLC** is OW-CCA secure. Significantly, assume there is a **Type-2 adversary** $\mathcal{A}_2$ that has advantages $\epsilon_2(\lambda)$ against the **PKE-ET-HS-CLC**. Assume that $\mathcal{A}_2$ makes q_{pk} public key queries, q_{sk} secret key queries, q_t trapdoor queries, q_{dec} decryption queries, q_{H_2} hash queries to H_2, and q_{H_3} hash queries to H_3. Thus, we have an algorithm $\mathcal{B}_2$ which breaks BDH problem with advantage at least $\epsilon_2(\lambda)/e(q_{sk} + q_t + 1).q_{H_3}$.

Proof: Presuming that $\mathcal{B}_2$ utilizes as inputs the BDH parameters $\langle p, \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and arbitrary instance $< g, g^a, g^b, g^c >$ of the BDH problem, where g is a random generator of $\mathbb{G}_1$ and $a, b, c \in_R \mathbb{Z}_p^*$. Suppose that $e(g, g)^{abc} \in \mathbb{G}_2$ is the solution of BDH problem. Then, we demonstrate how an algorithm $\mathcal{B}_2$ obtains $e(g, g)^{abc}$ by reacting with $\mathcal{A}_2$ as follows:

1. **Setup:** The algorithm $\mathcal{B}_2$ gives $\mathcal{A}_2$ the public parameters $PubP = \langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2, H_3\rangle$, where $g_1 = g^{s_1}, g_2 = g^{s_2}$. Then, $\mathcal{B}_2$ gives *PubP* and the master secret key *msk* $=(s_1, s_2)$ to $\mathcal{A}_2$.
2. **Phase 1:** At any time $\mathcal{A}_2$ can make queries to q_{pk}, q_{sk}, q_t, q_{dec}, q_{H_2}, or q_{H_3}. To respond to these queries $\mathcal{B}_2$ works as follows:
   - H_1-**queries**, H_2-**queries**, and H_3-**queries** are same as in **Phase 1** for proof of **Theorem 6.1.1**.
   - **Public key queries**(ID_i)**:** Algorithm $\mathcal{B}_2$ prepares a list of tuples $< ID_i, x_i, D_i, pk_i, sk_i, c_i >$ denoted by PSK^{list} and responds as follows:
     - o Check the H_1^{list}, if $c_i = 0$, $\mathcal{B}_2$ executes the *Assign secret value* algorithm to compute x_i, then calculates $D_i = (D_{i,1}, D_{i,2}) = (g_1^{z_i}, g_2^{z_i})$, $sk_i = (sk_{i,1}, sk_{i,2}) = (D_{i,1}^{x_i}, D_{i,2}^{x_i})$, and $pk_i = (X_i, pk_{i,1}, pk_{i,2}) = (g^{x_i}, g_1^{x_i}, g_2^{x_i})$ by executing *Extract partial secret key, Assign secret key,* and *Assign public key* algorithms, respectively. Finally, $\mathcal{B}_2$ records item $[ID_i, x_i, D_i, pk_i, sk_i, 0]$ into PSK^{list} and responds to $\mathcal{A}_2$ with pk_i.
     - o Otherwise, if $c_i = 1$, $\mathcal{B}_2$ executes the *Extract partial secret key* algorithm to compute $D_i = (D_{i,1}, D_{i,2}) = (g_1^{z_i}, g_2^{z_i})$, then calculates $pk_i = (X_i, pk_{i,1}, pk_{i,2}) = (g^a, g_1^a, g_2^a)$ by executing

Set public key algorithm, and records item $[ID_i, D_i, *, pk_i, *, 1]$ into PSK^{list} and responds to $\mathcal{A}_2$ with pk_i.

- **Secret key queries**(ID_i)**:** If $c_i = 0$, $\mathcal{B}_2$ returns sk_i associated with $ID_i$from PSK^{list}. Otherwise, if $c_i = 1$, $\mathcal{B}_2$ aborts and fails.
- **Decryption queries**(ID_i, C_i)**:** Let $C_i = (C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4})$. $\mathcal{B}_2$ searches on the PSK^{list} and reacts as follows:
    - If $c_i = 0$, $\mathcal{B}_2$ executes **CLC-Decrypt**(C_i, sk_i) and returns M_i to $\mathcal{A}_2$.
    - Otherwise, for each item in H_3^{list} $\mathcal{B}_2$ responds as follows:
        - Calculate $M_i || r_{i,1} = C_{i,4} \oplus h_i$.
        - Verify if $C_{i,1} = g^{r_{i,1}}$ and $\frac{C_{i,3}}{M_i^{r_{i,1}}} = H_2(e(sk_{i,1}, C_{i,1}))$.
        - If both verifications pass, returnM_i. Otherwise, return the symbol $\perp$.
- **Trapdoor queries**(ID_i)**:** $\mathcal{B}_2$ searches on PSK^{list} with respect to the ID_i and responds as follows:
    - If $c_i = 0$:
        - In case of **PKE-ET-HS-CLC**, $\mathcal{B}_2$ responds to the $\mathcal{A}_2$ with$td_{CLC_i} = g_1^{z_i x_i}$.
        - In case of **PKE-ET-HS-IBC**, $\mathcal{B}_2$ responds to the $\mathcal{A}_2$with $td_{IBC_i} = td_{IBC}$.
    - Else, if :
        - In case of **PKE-ET-HS-CLC**, $\mathcal{B}_2$ responds to the $\mathcal{A}_2$ with$td_{CLC_i} = g_1^{a z_i}$.
        - In case of **PKE-ET-HS-IBC**, $\mathcal{B}_2$responds to the $\mathcal{A}_2$ with $td_{IBC_i} = td_{IBC}$.

3. **Challenge:** When $\mathcal{A}_2$ decides **Phase 1** is finished. It outputs identity ID_{ch} on which it intends to be challenged on. The algorithm $\mathcal{B}_2$ chooses $M' \in_R \mathbb{G}_1^*$, then searches on PSK^{list} and reacts as follows:
    - If $C_i = 0$, terminate with failure.
    - Else, the following steps are performed:
        - Choose $C_4 \in_R \{0,1\}^{n_1+n_2}$ and set $C_2 = g^c$.
        - Generate $C' = (C_1, C_2, C_3, C_4)$ and send to $\mathcal{A}_2$ as a challenge ciphertext.
    - The decryption of C' will then be:

$$M^*||r = C_4 \oplus H_3(e(sk_{i,2}, C_2))$$
$$= C_4 \oplus H_3(e(g_2^{bz_i x_i}, g^c))$$

$$= C_4 \oplus H_3(e(g^{abz_i x_i}, g^c))$$
$$= C_4 \oplus H_3(e(g,g)^{abcz_i x_i}).$$

4. **Phase 2:** In this phase, the response of the challenger to $\mathcal{A}_2$ is similar to the one obtained in **Phase 1** on grounds that:
   - ID_{ch} is not queried in the *Secret key queries*.
   - (ID_{ch}, C^*) is not queried in the *Decryption queries*.
5. **Guess:** $\mathcal{A}_2$ returns M' for M^*. $\mathcal{B}_2$ picks item $[v_i, h_i] \in_R H_3^{list}$ and returns $h_i^{(zs_2)^{-1}} = e(g,g)^{abc}$ as the solution of BDH problem.

Claim: If $\mathcal{B}_2$ holds in the simulation, then $\mathcal{A}_2$ is viewed similar to real attack. Thus, the $Pr[M^* = M'] \geq \epsilon_2$.

Proof: H_1**-queries** responds as in the real attack. All replies to secret key queries q_{sk}, trapdoor queries q_t are valid. Then, the probability of $\mathcal{A}_1$ is $Pr[M = M'] \geq \epsilon_2$. Then, we compute the probability of $\mathcal{B}_2$'s failure in the simulation as follows:

1. The probability that $\mathcal{B}_2$ holds in **Phase 1** or **Phase 2** is equal to $\delta^{q_{sk}+q_{psk}+q_t}$.
2. The probability that $\mathcal{B}_2$ holds in **Challenge** phase is equal to $1-\delta$.
3. The combination of both gives the probability of $\mathcal{B}_2$ holds in the simulation is equal to $\delta^{q_{sk}+q_{psk}+q_t}(1-\delta)$.
4. The maximum of this probability is equal to $\delta_{opt} = 1 - 1/(q_{sk} + q_t + 1)$.
5. By using δ_{opt}, the probability that $\mathcal{B}_2$ holds is at least $1/e(q_{sk} + q_t + 1)$.

According to above analysis, the advantage of $\mathcal{B}_1$ is at least:

$$\epsilon_2(\lambda)/e(q_{sk} + q_t + 1) \quad (3)$$

In addition, the algorithm $\mathcal{B}_1$ is emulating the real attack environment to$\mathcal{A}_1$. Thus, $\mathcal{B}_1$ returns $e(g,g)^{abc}$ with probability at least:

$$\epsilon_2(\lambda)/q_{H_3}. \quad (4)$$

By combining Equations (3) and (4), we have

$$\epsilon_3(\lambda)/e(q_{sk} + q_t + 1).q_{H_3}.$$

The proof of **Theorem 6.1.2** is completed.

6. 2 PKE-ET-HS-IBC

Theorem 6.2.1: *Presuming that* H_1, H_2, H_3 *are random oracles and assume that the BDH problem is hard. Therefore, the* ***PKE-ET-HS-IBC*** *is OW-ID-CCA secure. Significantly, assume there is an OW-ID-CCA adversary* $\mathcal{A}$ *that has advantages* $\epsilon(\lambda)$ *against the* ***PKE-ET-HS-IBC****. Assume that* $\mathcal{A}$ *makes* q_E *key generation queries,* q_{td} *trapdoor queries, and* q_{H_3} *hash queries to* H_3*. Hence, we have an algorithm* $\mathcal{B}$ *which breaks BDH problem with advantage at minimum*$\epsilon(\lambda)/e(q_E + q_{td} + 1).q_{H_3}$.

Proof: To prove **Theorem 6.2.1**, we need to define the related public key encryption scheme *PubIB*, which will be used as tools in our proof. The *PubIB* is presented by three algorithms as follows:

1. **KeyGen**: By giving a security parameter λ, the algorithm performs as follows:
   - Create two groups $\mathbb{G}_1$, $\mathbb{G}_2$ with prime order *p*, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Then, pick a random generator $g \in \mathbb{G}_1$.
   - Pick two hash functions $H_2 : \mathbb{G}_2 \to \mathbb{G}_1$and $H_3 : \mathbb{G}_2 \to \{0, 1\}^{n_1+n_2}$.
   - Randomly choose $(s_1, s_2) \in \mathbb{Z}_p^*$, then set $g_1 = g^{s_1}$ and $g_2 = g^{s_2}$. Choose a random $h_{ID} \in \mathbb{G}_1$.
   - The $\langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, h_{ID}, H_2, H_3\rangle$ is a public key. The secret key $sk = (sk_1, sk_2) = (h_{ID}^{s_1}, h_{ID}^{s_2}) \in \mathbb{G}_1$.
2. **Encryption:** It takes the plaintext $M \in \mathbb{G}_1$ and the public key $\langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, h_{ID}, H_2, H_3\rangle$ as inputs. Then, it picks two numbers $(r_1, r_2) \in_R \mathbb{Z}_p^*$ and computes $C = (C_1, C_2, C_3, C_4)$, where$C_1 = g^{r_1}$, $C_2 = g^{r_2}$, $C_3 = M^{r_1}.H_2(e(h_{ID}, g_1)^{r_1})$, and $C_4 = (M||r_1) \oplus H_3(e(h_{ID}, g_2)^{r_2})$.
3. **Decryption:** It takes the secret key sk_{IBC} and the ciphertext *C* as inputs. Then, it returns the plaintext *M* by computing $M||r_1 \leftarrow C_4 \oplus H_3(e(sk_2, C_2))$, and then verifies both$C_1 = g^{r_1}$ and $\frac{C_3}{M^{r_1}} = H_2(e(sk_1, C_1))$. If both verifications pass, it returns *M*. Otherwise, it returns the symbol $\perp$.

After defining *PubIB*, we prove **Theorem.2.1** by using **Lemma 6.2.2** and **Lemma 6.2.3**, respectively as follows:

Lemma 6.2.2: *Presuming that* H_1 *is a random oracle. Suppose that the advantage of* $\mathcal{A}$ *against* ***PKE-ET-HS-IBC*** *is* $\epsilon(\lambda)$*. Assume that* $\mathcal{A}$ *makes* q_E *key generation queries,* q_{td} *trapdoor queries. There is an OW-CCA adversary* $\mathcal{B}$ *that has advantage* $\epsilon(\lambda)/e(q_E + q_{td} + 1)$ *against PubIB.*

Proof: We demonstrate the way of constructing an OW-CCA adversary $\mathcal{B}$ that utilizes $\mathcal{A}$ to get advantages $\epsilon(\lambda)/e(q_E + q_{td} + 1)$ against *PubIB*. Initially, $\mathcal{B}$ execute **KeyGen** algorithm to create the $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, h_{ID}, \mathcal{H}_2, \mathcal{H}_3 \rangle$ and a secret key $sk = (h_{ID}^{s_1}, h_{ID}^{s_2})$. Finally, $\mathcal{B}$ sends K_{pub} to $\mathcal{A}$. In the following, the algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ in the OW-ID-CCA game.

1. **Setup:** The algorithm $\mathcal{B}$ gives $\mathcal{A}$ the *PubIB* public parameters $\langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_1, H_2, H_3 \rangle$. Here, we take $p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, H_2, H_3$ from K_{Pub} and H_1 is a random oracle dominated by $\mathcal{B}$ as follows:

H_1-queries: $\mathcal{A}$ can query to H_1 at any time. The list of tuples $\langle ID_i, d_i, x_i, c_i \rangle$ denoted by H_1^{list} is prepared by $\mathcal{B}$ to respond to these queries. Initially, the H_1^{list} is blank. When $\mathcal{A}$ queries to H_1 with identity ID_i, $\mathcal{B}$ reacts as follows:

- If the ID_i exists in the H_1^{list}, the algorithm $\mathcal{B}$ returns $d_i = H_1(ID_i) \in \mathbb{G}_1$.
- Else, $\mathcal{B}$ generates the random coin $c_i = \{0, 1\}$, where 1 represents the probability of δ and 0 represents the probability of $1 - \delta$.
- $\mathcal{B}$ chooses a random number $x_i \in \mathbb{Z}_p^*$. If $c_i = 0$, $\mathcal{B}$ computes $d_i = g^{x_i}$. If $c_i = 1$, $\mathcal{B}$ computes $d_i = h_{ID}^{x_i} \in \mathbb{G}_1$.
- $\mathcal{B}$ adds tuples $\langle ID_i, d_i, x_i, c_i \rangle$ and responds to $\mathcal{A}$ with $d_i \in \mathbb{G}_1$.

2. **Phase 1:**
   - **Key generation queries** $\langle ID_i \rangle$: Algorithm $\mathcal{B}$ responds to this query as follows:
     - The above algorithm is run in response to H_1-queries in order to get $d_i \in \mathbb{G}_1$.
     - Let $\langle ID_i, d_i, x_i, c_i \rangle$ are the compatible tuples of the H_1^{list}. If $c_i = 1, \mathcal{B}$ aborts with failure. If $c_i = 0$, $d_i = g^{x_i}$, we define $sk_i = (g_1^{x_i}, g_2^{x_i}) \in \mathbb{G}_1$. Observe that $sk_i = ((g^{s_1})^{x_i}, (g^{s_2})^{x_i})$ and therefore sk_i is the secret key associated with ID_i. The algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with sk_i.
   - **Trapdoor query**: Algorithm $\mathcal{B}$ responds to this query as follows:
     - In case of **PKE-ET-HS-IBC**, $\mathcal{B}$ responds to $\mathcal{A}$ with $td_{IBC_i} = g_1^{x_i}$.
     - In case of **PKE-ET-HS-CLC**, $\mathcal{B}$ responds to $\mathcal{A}$ with td_{CLC_i}.
3. **Challenge:** When the algorithm $\mathcal{A}$ decides **Phase 1** is finished. It returns ID^* as the identity on which it wishes to be challenged. The algorithm $\mathcal{B}$ reacts as follows:
   - $\mathcal{B}$ runs the above algorithm in order to respond to the queries issued by H_1 to get $d \in \mathbb{G}_1$, where $\langle ID^*, d, x, c \rangle$ are the compatible tuples of the H_1^{list}.
   - If $c = 0$ then $\mathcal{B}$ aborts with failure. The attack on *PubIB* fails.

- If $c = 1$, thus $d = h^x_{ID}$. Remember that once $C = (C_1, C_2, C_3, C_4)$, we have $(C_1, C_2) \in \mathbb{G}^*_1$. Set $C^* = (C_1^{x^{-1}}, C_2^{x^{-1}}, C_3, C_4)$, where x^{-1} is the inverse of $x \bmod \text{p}$. $\mathcal{B}$ returns C^* to $\mathcal{A}$ as the challenge ciphertext, where C^* is the result of the plaintext encryption *M* under the challenge identity ID^*. The ciphertext is legal because $H_1(ID^*) = d$, the decryption key associated with ID^* is $sk^*_{IBC} = (sk^*_1, sk^*_2) = ((h^x_{ID})^{s_1}, (h^x_{ID})^{s_2})$.

$$e(sk_{ch,1}, C_1^{x^{-1}}) = e((h^x_{ID})^{s_1}, C_1^{x^{-1}}) = e(h^{s_1}_{ID}, C_1).$$
$$e(sk_{ch,2}, C_2^{x^{-1}}) = e((h^x_{ID})^{s_2}, C_2^{x^{-1}}) = e(h^{s_2}_{ID}, C_2).$$

- Thus, the **PKE-ET-HS-IBC** decryption of C^* using sk^*_{IBC} is identical to the *PubIB* decryption of C using sk_{IBC_i}.

4. **Phase 2:**
   - **Key generation queries** $\langle ID_i \rangle$: If $ID_i \neq ID^*$, $\mathcal{B}$ responds similar to **Phase 1**. Otherwise, $\mathcal{B}$ aborts with failure.
   - **Decryption queries** $\langle ID_i, C_i \rangle$: If $C_i \neq C^*$, $\mathcal{B}$ responds the same way as in **Phase 1**. Otherwise, Otherwise, $\mathcal{B}$ aborts with failure.
   - **Trapdoor queries**: $\mathcal{B}$ responds the same way as in **Phase 1**.
5. **Guess:** Finally, $\mathcal{A}$ outputs a guess M' for M. $\mathcal{B}$ outputs a guess M' as its guess for *M*.

Claim: If $\mathcal{B}$ holds in the simulation, then $\mathcal{A}$ is viewed similar to real attack. Thus, the $Pr[M = M'] \geq \epsilon$.

Proof: H_1-queries responds as in the real attack. All replies to key generation queries q_E and trapdoor queries q_{td} are valid. Then, the probability of $\mathcal{A}$ is $Pr[M = M'] \geq \epsilon$.

To finalize the proof of **Lemma 6.2.2**, we compute the possibility of $\mathcal{B}$'s failure in the simulation as follows: (1) The possibility of $\mathcal{B}$ holds in **Phase 1** or **Phase 2** is equal to $\delta^{q_E + q_{td}}$.(2) The possibility of $\mathcal{B}$ holds in **Challenge** phase is equal to $1 - \delta$. (3) The combination of both gives the possibility of $\mathcal{B}$ holds in the simulation is equal to $\delta^{q_E + q_{td}}(1 - \delta)$. (4) The maximum of this possibility is equal to $\delta_{opt} = 1 - 1/(q_E + q_{td} + 1)$. By using δ_{opt}, the possibility of $\mathcal{B}$ holds is at minimum $1/e(q_E + q_{td} + 1)$. This result shows that the advantage of $\mathcal{B}$ is at minimum:

$$\epsilon(\lambda)/e(q_E + q_{td} + 1). \tag{1}$$

Lemma 5.2.3: Assume that H_3 is a random oracle $H_3 : \mathbb{G}_2 \to \{0,1\}^{n_1+n_2}$. Assume that the adversary $\mathcal{P}$ is an OW-CCA adversary with advantage $\epsilon(\lambda)$ against *PubIB*. Assume that $\mathcal{P}$ makes at most q_{H_3} hash queries of H_3. Thus, the algorithm $\mathcal{F}$ solves the BDH assumption with advantage at minimum $\epsilon(\lambda)/q_{H_3}$.

Proof: Algorithm $\mathcal{F}$ takes BDH parameters $\langle p, \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and the tuple $\langle g_0, g_0^a, g_0^b, g_0^c\rangle = \langle g_0, V, W, R\rangle$, where g_0 is the generator of $\mathbb{G}_1$ and $a, b, c \in_R Z_p^*$ as inputs. Let $S = e(g_0, g_0)^{abc} \in \mathbb{G}_2$ be the solution of BDH. In the following, we show how the algorithm $\mathcal{F}$ finds S by interacting with $\mathcal{P}$.

Setup: $\mathcal{F}$ gives $\mathcal{P}$ the *PubIB*'s public parameters $K_{pub} = \langle p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, g_2, h_{ID}, H_2, H_3\rangle$. Then $\mathcal{F}$ sets $g_2 = V$ and $h_{ID} = W$. Here H_3 is the random oracle dominated by $\mathcal{F}$. Observe that a decryption key related to K_{pub} is $sk_i = h_{ID}^a = g_0^{ab}$.

H_3**-queries:** At any time $\mathcal{P}$ can query to H_3. $\mathcal{F}$ prepares the list $\langle X_i, H_i\rangle$ denoted by H_3^{list} to respond these quries. Initially, the H_3^{list} is blank. When the $\mathcal{P}$ query to H_3, $\mathcal{F}$ responds as follows:

1. If X_i appears in the H_3^{list}, the $\mathcal{F}$ returns $H_3(X_i) = H_i$.
2. Else, $\mathcal{F}$ chooses $H_i \in_R \{0,1\}^{n_1+n_2}$ and records it in H_3^{list}. Finally, $\mathcal{F}$returns $H_3(X_i) = H_i$ to $\mathcal{P}$.

Challenge: $\mathcal{P}$ sends the identity ID^* on which it wishes to be challenged. $\mathcal{F}$chooses $L \in_R \{0,1\}^{n_1+n_2}$ and generates C^* as a challenge ciphertext $C^* = (C_1, R, C_3, L)$. $\mathcal{F}$ sends C^* to $\mathcal{P}$. Observe that the decryption of C^* is $L \oplus H_3(e(sk_i, R)) = L \oplus H_3(S)$.

Guess: $\mathcal{P}$ outputs a guess M' for M. Algorithm $\mathcal{F}$ chooses $\langle X_j, H_j\rangle$ from the H_3^{list} randomly and returns X_j as the solution of the given BDH parameters.

The algorithm $\mathcal{F}$ is emulating the real attack environment to $\mathcal{P}$. Therefore, $\mathcal{F}$ returns S with probability at minimum:

$$\epsilon(\lambda)/q_{H_3}. \quad (5)$$

By combining Equations (5) and (6), we have

$$\epsilon(\lambda)/e(q_E + q_{td} + 1).q_{H_3}. \quad (6)$$

The proof of Theorem 6.2.1 is completed.

7. Performance analysis

Within this segment, the computation and communication costs of the proposed **PKE-ET-HS** scheme and the schemes in [14][16], and [21] are compared.

Table 1. Comparison.

		[14]-Type 1	[16]	[21]	Ours
Comp of	ENC	6Exp	2Pair+6Exp	4Pair+5Exp	2Pair+5Exp
	DEC	5Exp	2Pair+2Exp	2Pair+2Exp	2Pair+2Exp
	Test	2Pair+2Exp	4Pair	4Pair	4Pair
Size of	PK	$3\mathbb{G}$	$2\mathbb{G}$	$2\mathbb{G}$	$2\mathbb{G}$
	SK	$3\mathbb{Z}_p$	$2\mathbb{Z}_p$	$2\mathbb{Z}_p$	$2\mathbb{Z}_p$
	CT	$5\mathbb{G}+\mathbb{Z}_p$	$4\mathbb{G}+\mathbb{Z}_p$	$5\mathbb{G}+\mathbb{Z}_p$	$3\mathbb{G}+\mathbb{Z}_p$
Property	Security	OW-CCA	OW-ID-CCA	OW-CCA	OW-CCA & OW-ID-CCA
	Assumption	CDH	BDH	BDH	BDH
	Heterogeneous (ET)	×	×	×	✓
	CMP	✓	×	×	×

Legends: Enc, Dec, and Test: the computation complexity of encryption, decryption, and test algorithms; PK, SK, and CT: the size of public key, the size of secret key, and the size of ciphertext, respectively; BDH: bilinear Diffie-Hellman assumption; CDH: computational Diffie-Hellman assumption; Exp: an exponentiation operation; Pair: pairing operation; heterogeneous (ET): the scheme that provides heterogeneous equality test; CMP: certificate management problems; ×: this property is not considered in the corresponding scheme; ✓: this property is considered in the corresponding scheme.

1. Size and storage space:
    - Public key: Our scheme has the same size as [16] and [21]. It's smaller than [14]-Type 1.
    - Secret key: Our scheme has the same size as [16] and [21]. It's smaller than [14]-Type 1.
    - Ciphertext: Our scheme has the smallest ciphertext size.
2. Computation complexity:
    - Encryption algorithm: Our scheme has less computational cost than [16] and [21]. It's larger than [14]-Type 1.

- Decryption algorithm: Our scheme has same computational cost with [16] and [21]. It's larger than [14]-Type 1.
- Test algorithm: Our scheme has same computational cost with [16] and [21]. It's larger than [14]-Type 1.

3. Security: Our scheme and the scheme in [16][21] are proved under the BDH problem. Besides, the scheme in [10] is proved under the CDH problem.
4. Assumption: Our scheme achieves OW-ID-CCA and OW-CCA security. Besides, the scheme in [14][21] are secure in OW-CCA and the scheme in [16] achieves OW-ID-CCA security.
5. Heterogeneous (ET): Only our schemes provide heterogeneous equality test between CLC cryptosystem and IBC cryptosystem. Beside, all of others provide homogenous equality test.
6. Certificate management problems: Our scheme and the schemes in [16][21] are solved the certificate management problems.

To intuitively evaluate theoretical analysis illustrated above, based on cpabe toolkit and Pairing-Based Cryptography (PBC) library [26], our scheme and the schemes in [14][16][21] are implemented. Particularly, these experiments are carried out on an Intel(R) Core(TM) i7-7700 CPU @3.60 GHz @3.60 GHz and 8 GB RAM. We used the Windows 10 operating system and VC++ 6.0 for our experiments to run on. We implemented the 160-bit elliptic curve group and constructed it on the super-singular curve $z^2 = x^3 + x$ over a 512-bit finite field, to attain a 1024-bit security level. For the cost of computation and communication overhead, we had milliseconds (ms) and byte, respectively. In our experimental simulation, the time of the pairing operation and the exponentiation operation are 10.749 ms and 5.530 ms, respectively. Besides, we have the size of each element in $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$ are 20 bytes, 128 bytes, 128 bytes, respectively.

Based on the results we obtained during the experimental process, **Fig. 3** demonstrates that the computation cost of our nominated scheme is moderate as contrasted to the schemes in [16][21] and it's higher than the schemes in [14]. Furthermore, **Fig. 4** demonstrates that our proposed scheme has a moderate communication cost in contrast to schemes in [14][16][21].

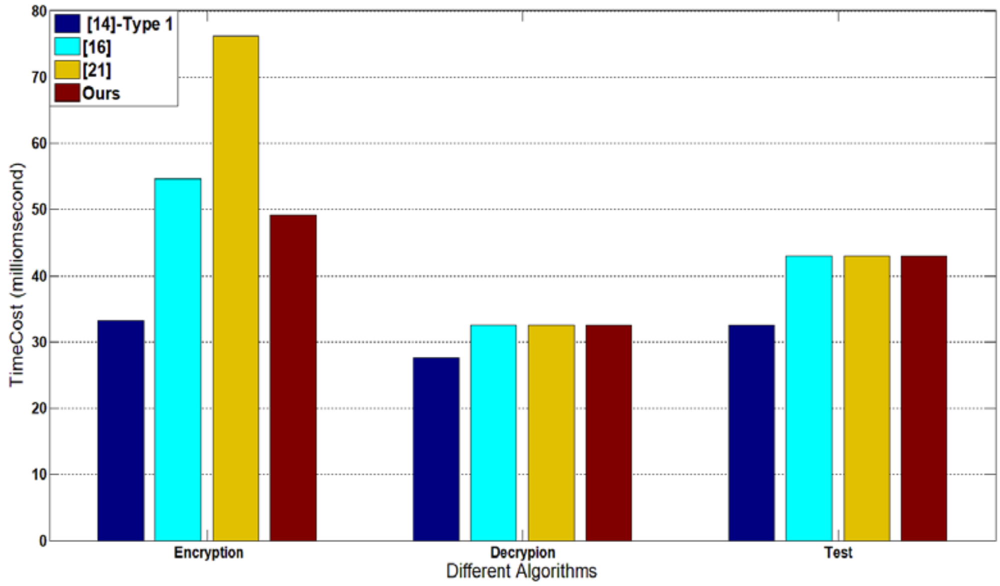

Fig. 3. Computation costs for Encryption, Decryption, and Test algorithms

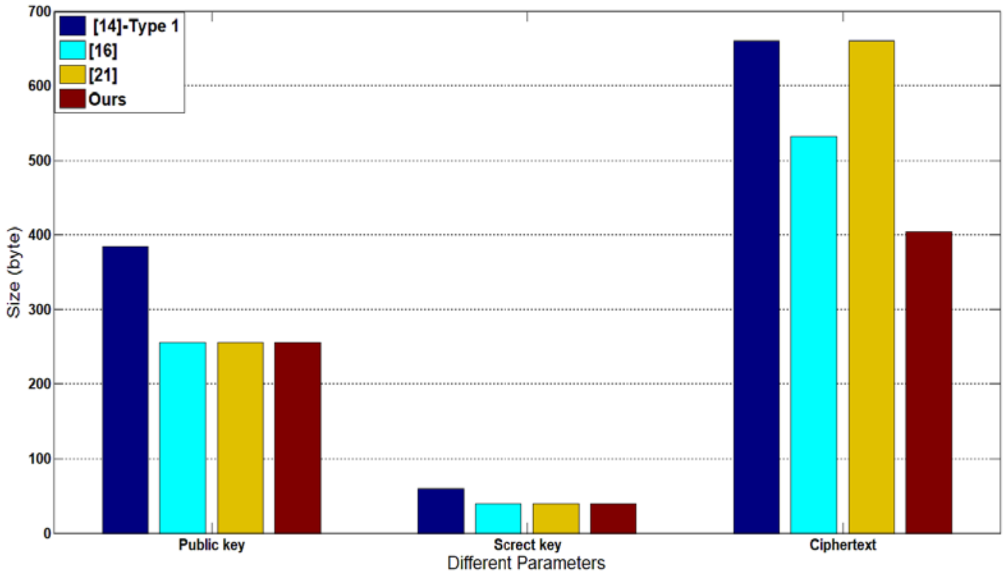

Fig. 4. Comparison of the communication cost

8. Conclusion

In this paper, we propose a novel public key encryption with equality test in heterogeneous systems. This scheme is aimed at dealing with the cloud server practical needs. We have come up with a mechanism to allow a cloud server execute a search between ciphertexts encrypted amidst the CLC cryptosystem and IBC cryptosystem. Our scheme has its security proof reduced to Bilinear Diffie-Hellman assumption. We base this scheme on the random oracle model. According to the analysis and simulations we undertook, our experiments reveal that our scheme is feasible and sufficient in correlation to other works. Forthcoming works include making provision for the cloud server to be delegated with rights to execute authorization tests.

Acknowledgments

This work was supported in part by the 13th Five-Year Plan of National Cryptography Development Fund for Cryptographic Theory of China under Grant MMJJ20170204, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2016J091, the Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, and in part by the Natural Science Foundation of China under Grants U1401257, 61472064 and 61602096.

References

[1] N. Fernando, S. W. Loke, W. Rahayu, "Mobile cloud computing: A survey," *Future generation computer systems*, 29 (1), 84–106, 2013.

[2] A. Botta, W. De Donato, V. Persico, A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future Generation Computer Systems*, 56, 684–700, 2016.

[3] Q. Yan, F. R. Yu, Q. Gong, J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, 18(1), 602–622, 2016. Article (CrossRef Link).

[4] H. Takabi, J. B. Joshi, G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, 8(6), 24–31, 2010. Article (CrossRef Link).

[5] H. Hong, Z. Sun, "Sharing your privileges securely: a key-insulated attribute based proxy re-encryption scheme for iot," *World Wide Web*, 21(3), 595–607, 2018. Article (CrossRef Link).

[6] H. Hong, Z. Sun, "Achieving secure data access control and efficient key updating in mobile multimedia sensor networks," *Multimedia Tools and Applications*, 77(4), 4477–4490, 2018. Article (CrossRef Link).

[7] H. Hong, Z. Sun, "A key-insulated ciphertext policy attribute based signcryption for mobile networks," *Wireless Personal Communications*, 95(2), 1215–1228, 2017. Article (CrossRef Link).

[8] H. Hong, X. Liu, Z. Sun, "A fine-grained attribute based data retrieval with proxy re-encryption scheme for data outsourcing systems," *Mobile Networks and Applications*, 1–6, 2018. Article (CrossRef Link).

[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, "Public key encryption with keyword search," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques, Springer*, pp. 506–522, 2004. Article (CrossRef Link).

[10] G. Yang, C. H. Tan, Q. Huang, D. S. Wong, "Probabilistic public key encryption with equality test," in *Proc. of Cryptographers Track at the RSA Conference, Springer*, pp. 119–131, 2010. Article (CrossRef Link).

[11] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," *Security and Communication Networks*, 5(12), 1351–1362, 2012. Article (CrossRef Link).

[12] S. Ma, M. Zhang, Q. Huang, B. Yang, "Public key encryption with delegated equality test in a multi-user setting," *The Computer Journal*, 58(4), 986–1002, 2014. Article (CrossRef Link).

[13] K. Huang, R. Tso, Y.-C. Chen, S. M. M. Rahman, A. Almogren, A. Alamri, "Pke-aet: public key encryption with authorized equality test," *The Computer Journal*, 58(10), 2686–2697, 2015.

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

[15] Y. Xu, M. Wang, H. Zhong, J. Cui, L. Liu, V. N. Franqueira, "Verifiable public key encryption scheme with equality test in 5g networks," *IEEE Access 5*, 12702–12713, 2017. Articale (Cross Ref Link).

[16] S. Ma, "Identity-based encryption with outsourced equality test in cloud computing," *Information Sciences*, 328, 389–402, 2016. Articale (Cross Ref Link).

[17] H. T. Lee, S. Ling, J. H. Seo, H. Wang, "Semi-generic construction of public key encryption and identity-based encryption with equality test," *Information Sciences*, 373, 419–440, 2016. Articale (Cross Ref Link).

[18] L. Wu, Y. Zhang, K.-K. R. Choo, D. He, "Efficient and secure identity-based encryption scheme with equality test in cloud computing," *Future Generation Computer Systems*, 73, 22–31, 2017.

[19] L. Wu, Y. Zhang, K.-K. R. Choo, D. He, "Efficient identity-based encryption scheme with equality test in smart city," *IEEE Transactions on Sustainable Computing*, 3(1), 44–55, 2018.

[20] S. S. Al-Riyami, K. G. Paterson, "Certificateless public key cryptography," *Asiacrypt*, *Springer*, Vol. 2894, pp. 452–473, 2003.

[21] H. Qu, Z. Yan, X.-J. Lin, Q. Zhang, L. Sun, "Certificateless public key encryption with equality test," *Information Sciences*, 462, 76–92, 2018. Article (CrossRef Link).

[22] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," *Crypto, Springer*, 2(3), 350-391, 2008. Article (CrossRef Link).

[23] Q. Tang, "Towards public key encryption scheme supporting equality test with fine-grained authorization," in *Proc. of Australasian Conference on Information Security and Privacy, Springer*, pp. 389–406, 2011.

[24] Q. Tang, "Public key encryption schemes supporting equality test with authorization of different granularity," *International journal of applied cryptography*, 2(4), 304–321, 2012. Article (CrossRef Link).

[25] D. Boneh, M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. of Annual International Cryptology Conference, Springer*, pp. 213–229, 2001. Article (CrossRef Link).

[26] B. Lynn et al., "The pairing-based cryptography library," 2006. [Online]. Available: crypto. stanford. edu/pbc/[Mar. 27, 2013]

Rashad Elhabob received the B.S. degree, in 2010 from the Faculty of Computer Science and Information Technology, University of Karary, Khartoum, Sudan. In 2014 received his Master degree from Faculty of Mathematical Science, University of Khartoum, Khartoum, Sudan. He is currently pursuing the Ph.D. degree with the School of Software Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include cryptography and network security.

Yanan Zhao is currently pursuing her M.S. degree from the School of Information and Software Engineering, University of Electronic Science and Technology of China. She received her B.S. degree from Jiangxi University of Science and Technology in 2017. Her research interests include identity-based public key cryptography.

Iva Sella is currently pursuing her M.S. degree from the School of Information and Software Engineering, University of Electronic Science and Technology of China. She received her BTech (IT) degree from the Technical University of Kenya in 2014. Her research interests include certificateless public key cryptography.

Hu Xiong received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2009. He is currently a Full Professor with the UESTC. His research interests include public key cryptography and networks security.