electronics

MDPI

Article

Certificateless Encryption Supporting Multi-Ciphertext Equality Test with Proxy-Assisted Authorization

Siyue Dong [1,*,†,‡], Zhen Zhao [1,*,†,‡], Baocang Wang [1], Wen Gao [2] and Shanshan Zhang [3]

[1] The State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China; bcwang@xidian.edu.cn
[2] School of Cyberspace Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China; gaowen@xupt.edu.cn
[3] School of Mathematics and Information Science, Baoji University of Arts and Science, Xi'an 721016, China; sszhang@bjwlxy.edu.cn
* Correspondence: sydong@stu.xidian.edu.cn (S.D.); zzhen@xidian.edu.cn (Z.Z.)
† Current address: No. 2 South Taibai Road, Xi'an 710071, China.
‡ These authors contributed equally to this work.

Abstract: Public key encryption with equality test (PKEET) is a cryptographic primitive that enables a tester to determine, without decryption, whether two ciphertexts encrypted with different public keys generate from the same message. In previous research, public key encryption with equality test (PKEET) was extended to include identity-based encryption with equality test (IBEET), thereby broadening the application of PKEET. Subsequently, certificateless encryption with equality test (CLEET) was introduced to address the key escrow problem in IBEET. However, existing CLEET schemes suffer from inefficiency and potential information leakage when dealing with multiple ciphertexts due to the need for pairwise equality tests. To address this issue, we propose a concept of certificateless encryption supporting multi-ciphertext equality test with proxy-assisted authorization (CLE-MET-PA). CLE-MET-PA incorporates the functionality of the multi-ciphertext equality test into CLEET, enabling a tester to perform a single equality test on multiple ciphertexts to determine whether the underlying plaintexts are equal, without revealing any additional information. This enhances the security of our scheme while significantly reducing the computational overhead compared to multiple pairwise equality tests, making our scheme more efficient. Additionally, our approach integrates proxy-assisted authorization, allowing users to delegate a proxy to grant authorizations for equality tests on their behalf when offline. Importantly, the proxy token used in our scheme does not include any portion of the user's private key, providing enhanced protection compared to traditional PKEET schemes in which the user token is often part of the user's private key. We construct a concrete CLE-MET-PA scheme and prove that it achieves CPA security and attains CCA security through an FO transformation.

Keywords: public key encryption with equality test; certificateless encryption; multi-ciphertext equality test

Citation: Dong, S.; Zhao, Z.; Wang, B.; Gao, W.; Zhang, S. Certificateless Encryption Supporting Multi-Ciphertext Equality Test with Proxy-Assisted Authorization. *Electronics* **2023**, *12*, 4326. https://doi.org/10.3390/electronics12204326

Academic Editors: Weiting Zhang, Chuan Zhang and Tong Wu

Received: 7 October 2023
Revised: 12 October 2023
Accepted: 16 October 2023
Published: 18 October 2023

1. Introduction

Cloud computing provides an efficient solution for transferring the storage and computation burdens from users to cloud servers. In recent decades, the continuous improvement in network bandwidth, the emergence of big data, the Internet of Things, and artificial intelligence have led to the rapid development of cloud technology and its applications [1–4]. Cloud storage, in comparison to traditional local storage, offers several advantages including low cost, high scalability, easy management, and maintenance. With cloud technology, users can select the appropriate storage capacity and service types based on their specific requirements and perform various data operations such as uploading, downloading, backup, archiving, and sharing. As a result, more and more individuals and organizations are utilizing cloud

servers to process and store their data. However, due to the sensitive nature of their data and the need for privacy protection, it is common practice to encrypt the data on cloud servers using various cryptosystems. Nevertheless, there is a growing demand for cryptographic technologies that enable processing operations, such as searching, deduplication, classification, and data segmentation, while preserving the confidentiality of the encrypted data.

In the realm of processing encrypted data, researchers have proposed various cryptographic primitives, including searchable encryption [5,6] and fully homomorphic encryption [7]. Expanding upon searchable encryption, Boneh et al. [8] introduced public key encryption with keyword search (PEKS) in 2004. The scheme allows a server to verify, without decryption, whether a ciphertext C is generated from a message M as long as it is in possession of the corresponding tag T_M for that ciphertext and the public key used for message encryption. This property renders it suitable for applications such as the classification of public key ciphertexts.

This scheme, however, has a limitation in that it only allows ciphertext management for a single user, restricting comparison and search to ciphertexts under the same public key. To enable ciphertext equality tests across different public keys, additional primitives are necessary. In 2010, Yang et al. [9] proposed the concept of public-key encryption with equality test (PKEET). In their work, the authors introduced a special public key encryption scheme where an entity can determine whether two ciphertexts correspond to identical plaintexts using an equality test algorithm on two ciphertexts encrypted under different public keys.

1.1. Related Work

In the first PKEET scheme proposed by Yang et al. [9], there is a concern regarding the protection of data privacy for the data owner, as there is no restriction on who can perform an equality test. Consequently, the initial advancements in PKEET schemes were predominantly directed towards achieving fine-grained control over tester authorizations. These efforts [10–14] concentrated on devising methods to authorize testers and defining which ciphertexts they could perform equality tests on once granted authorization. The work proposed by Ma et al. [14] extensively examines and provides a comprehensive overview of diverse authorization mechanisms.

In 2016, Ma [15] and Lee et al. [16] introduced the identity-based encryption (IBE) primitive into PKEET, presenting their respective IBEET schemes. These schemes adopt the benefits of identity-based encryption (IBE) cryptosystems, which allow users to generate public and private key pairs using their identity information without the need for digital certificates, therefore addressing the key management problem in public key encryption cryptosystems. This novel approach sparked significant subsequent research on IBEET [17–28].

Certificateless public key encryption with equality test (CL-PKEET). When it comes to solving the key management problem, identity-based encryption (IBE) relies on a central authority, typically the key generation center (KGC), to generate private keys for users based on their identities. However, this reliance on a central authority means that the KGC has the ability to decrypt messages intended for any user. If the KGC is compromised, coerced, or misused, it can result in unauthorized access to users' encrypted data, which is known as the key escrow problem. To address this issue, Al-Riyami et al. [29] proposed certificateless encryption (CLE). In CLE, the KGC generates partial private keys for users, enabling them to generate their own private and public keys. This approach effectively addresses the key escrow problem. Similarly, IBEET also faces the key escrow problem. In order to tackle this problem in IBEET, Qu et al. [30] introduced CLE into PKEET, creating the initial certificateless public key encryption with equality test (CL-PKEET, referred to as CLEET in this paper for brevity) scheme.

Public key encryption with multi-ciphertext equality test (PKE-MET). Before the introduction of PKE-MET by Susilo et al. [31], all PKEET and IBEET schemes only supported pairwise equality tests. Consider a scenario where a tester receives n ciphertexts from n different users. With a conventional PKEET scheme, the tester would be required to perform

a minimum of $n-1$ equality tests between pairs of ciphertexts to determine the equality of the n ciphertexts. However, in the PKE-MET scheme, a specific parameter s is designated, enabling the tester to test the equality of s ciphertexts in a single equality test. Moreover, this feature of multi-ciphertext comparison significantly reduces additional information leakage in equality tests. For example, when conducting an equality test on ciphertexts from three different users to determine whether the plaintexts corresponding to user A, B, and C's ciphertexts are equal, traditional PKEET test algorithms would inevitably reveal additional information, such as the equality of the underlying plaintexts between user A and user B. By exploiting our multi-ciphertext test feature, the leakage of additional information in equality tests can be greatly mitigated. This improvement leads to better efficiency, heightened security, and reduced computational costs for the tester.

Proxy-assisted authorization. One authorization method employed in our scheme is consistent with the majority of PKEET schemes, known as user authorization. In this method, a user generates a token (also referred to as a trapdoor), and sends it to the tester, enabling the tester to conduct equality tests on the user's ciphertexts. Additionally, we introduce a proxy-assisted authorization method, whereby the user can engage with a proxy through a secure and efficient key exchange algorithm to create a proxy token. This token is then securely stored by the proxy and possesses the same authorization capabilities as the user token. There are two primary advantages to this approach. Firstly, it liberates the user from having to be constantly online to authorize a tester, enhancing the practical application value of the scheme. Secondly, unlike the user token, the proxy token does not compromise any part of the user's private key, thus reducing the risk of privacy exposure.

1.2. Our Contribution

We propose a CLE-MET-PA scheme in this paper, and the contributions of this paper are as follows:

- We introduce the multi-ciphertext equality test into CLEET. Our proposal associates each ciphertext with a designated number s, making it possible to perform equality tests on multiple ciphertexts simultaneously in a single test without revealing any additional information, all while retaining the fundamental attributes of certificateless encryption.
- We incorporate the concept of a proxy into our framework. Users have the flexibility to choose and disclose proxies along with their public keys on the public key server. This enables users to delegate authorization to proxies, allowing them to go offline, which effectively improves the practical application value. Moreover, the use of proxy tokens eliminates the exposure of the user's private key, thus enhancing the security of our scheme. Additionally, our encryption process does not involve proxy information. Hence, when users choose a new proxy, there is no need to reconfigure previous ciphertexts. The disclosed proxy information is only utilized in the equality test, resulting in a more flexible scheme. Furthermore, the key generation algorithm for proxies is identical to that of users, meaning that any user can act as a proxy. This enhances the flexibility and efficiency of our scheme.
- We establish formal security models for our concrete CLE-MET-PA scheme, including five different types of adversaries. Subsequently, our work achieves IND-CPA security against adversaries with the trapdoor of the challenge ciphertext and OW-CPA security against adversaries without the trapdoor of the challenge ciphertext. In the extension section, we employ the Fujisaki–Okamoto (FO) transformation [32,33] to modify the encryption and decryption processes, thereby attaining CCA security for our scheme (i.e., IND-CCA security against adversaries with the trapdoor of the challenge ciphertext and OW-CCA security against adversaries without the trapdoor of the challenge ciphertext).

1.3. Organization

In Section 2, we reviewed the concepts of asymmetric bilinear groups and the BDH assumption; in Section 3, we introduced the system model, definitions and security models of CLE-MET-PA; in Section 4, we presented the concrete construction of CLE-MET-PA and verified its correctness; In Section 5, we proved the security of the scheme; in Section 6, we conducted a performance analysis and discussed the security extension of the scheme; in Section 7, we summarized our work.

2. Preliminary

2.1. Asymmetric Bilinear Groups

Let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear groups ensemble, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three multiplicative cyclic groups of order p. $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ are two generators. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following properties:

1. *Bilinear:* For any $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. *Non-degenerate:* $e(g_1, g_2) \neq 1$.
3. *Computable:* For any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, there is an efficient algorithm to compute $e(g_1, g_2)$.

Since $\mathbb{G}_1 \neq \mathbb{G}_2$, this is an asymmetric bilinear map. In the asymmetric setting, if there exists an efficient computable isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$, the pairing e is referred to as a *Type 2 pairing*. Conversely, if no such isomorphism exists, e is categorized as a *Type 3 pairing* [34–37].

2.2. Bilinear Diffie–Hellman (BDH) Assumption in Asymmetric Bilinear Groups

The bilinear Diffie–Hellman (BDH) assumption was initially proposed by Boneh et al. [38] in the context of symmetric bilinear groups. Later, Boyen et al. [39] expanded this assumption to asymmetric bilinear settings.

The BDH problem can be defined as follows: consider a set of bilinear groups in Section 2.1, denoted by $\mathcal{G} = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\}$. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be two generators. Given an instance $(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b) \in \mathbb{G}_1^3 \times \mathbb{G}_2^3$ where a, b, and c are uniformly random choices from $\mathbb{Z}_p^*$, the task is to compute $e(g_1, g_2)^{abc} \in \mathbb{G}_T$.

The BDH assumption is defined as follows: Given an instance of the aforementioned BDH problem, no probabilistic polynomial-time (PPT) adversary, denoted by $\mathcal{A}$, can compute the value of $e(g_1, g_2)^{abc}$ with non-negligible probability. The advantage of $\mathcal{A}$ is defined as:

$$Adv(\mathcal{A}) = Pr[\mathcal{A}(g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b) = e(g_1, g_2)^{abc}].$$

3. System Model, Definitions, and Security Models

We present the system model of our certificateless encryption supporting multi-ciphertext equality test with proxy-assisted authorization (CLE-MET-PA) scheme, followed by its formal definition and security models.

3.1. System Model of CLE-MET-PA

The system model of our CLE-MET-PA scheme is depicted in Figure 1. The model comprises seven types of entities as follows:

1. Key generation center (KGC): This entity is responsible for system setup, safeguarding the master secret key, and issuing partial private keys to users based on their identities.
2. Message sender: This entity generates ciphertexts using the public key of the target user and uploads them to the cloud server.
3. Message receiver: This entity, often referred to as the user in this paper, can download ciphertexts for decryption, grant authorization to testers for equality tests, and delegate proxies to provide authorizations on its behalf.

4. Cloud server: This entity stores the ciphertexts generated by message senders and allows message receivers to download them. The cloud server often serves as the tester. To ensure generality, we established a separate entity for test purposes.
5. Proxy: This entity can interact with a message receiver to create a proxy token, granting authorization for equality tests on the message receiver's ciphertexts.
6. Public key server: This entity stores public keys issued by message receivers and proxies. Additionally, it keeps track of the message receiver's choice of proxy and publishes its proxy information.
7. Tester: This entity can perform an equality test on a set of s ciphertexts. To conduct the test, the tester must receive s ciphertexts along with the tokens issued by their respective message receivers or proxies. Each ciphertext is designated with the number s.

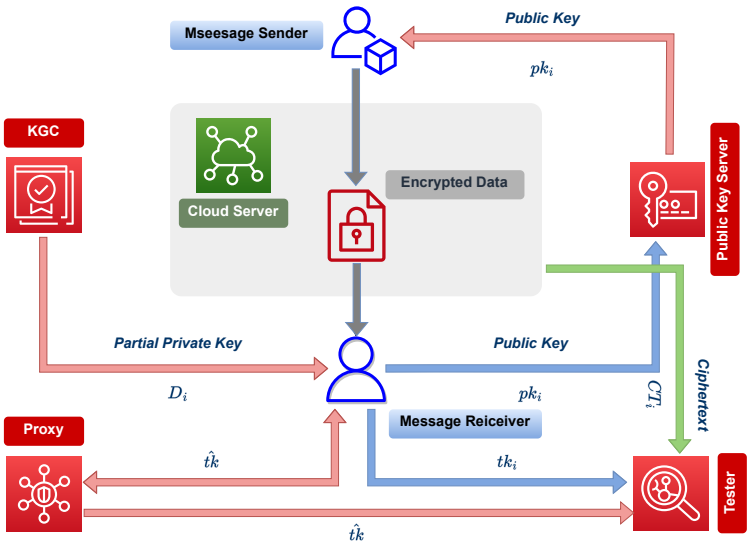

Figure 1. System model of CLE-MET-PA.

The workflow of our proposed CLE-MET-PA scheme is also depicted in Figure 1. The message sender queries the public key server for the necessary public key, encrypts the message using it to obtain the ciphertext, and uploads it to the cloud server. The message receiver can then download the ciphertext from the cloud server, and decrypt it to obtain the message. A message receiver can authorize a tester directly or by a delegated proxy, enabling the tester to perform an equality test on its ciphertexts.

A notable feature of CLE-MET-PA is its multi-ciphertext equality test, illustrated in Figure 2. A tester can conduct a single-test algorithm upon receiving s ciphertexts, each with the designated number s, along with s trapdoors (whether authorized by the user or authorized through user-delegated proxy authorization). The test algorithm outputs 1 if all underlying plaintexts to the ciphertexts are equal, and 0 if at least one underlying plaintext differs from the rest.

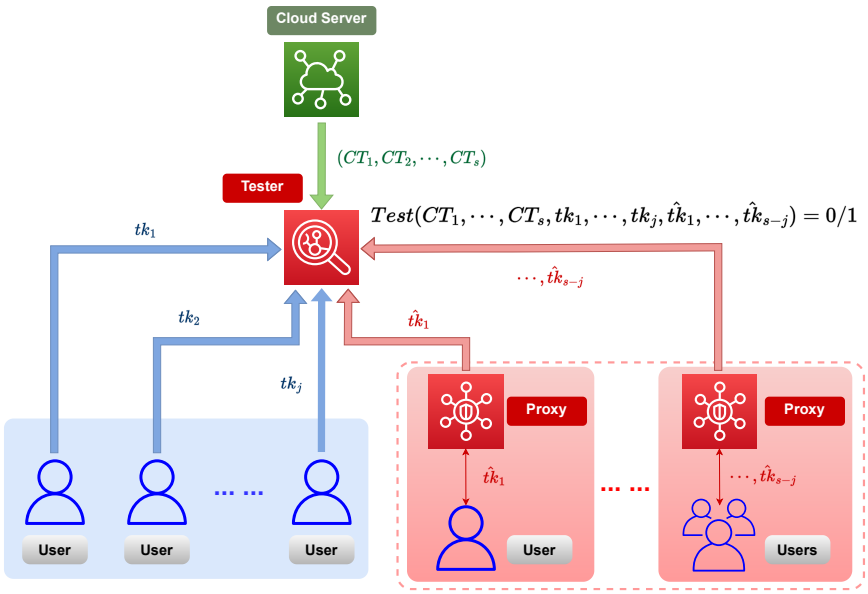

Figure 2. Equality test of CLE-MET-PA.

3.2. Certificateless Encryption Supporting Multi-Ciphertext Equality Test with Proxy-Assisted Authorization

A CLE-MET-PA system consists of eleven algorithms as described below:

- Setup(1^λ): This algorithm takes the security parameter λ as input and outputs the system parameter pp and the system master secret key msk.
- Partial-Private-Key-Extract(pp, msk, ID): Given the system parameter pp, a master key msk, and an identifier ID, this algorithm outputs the partial private key D.
- Set-Secret-Value (pp, ID): Given the system parameter pp and an identifier ID, this algorithm outputs a secret value x at random.
- Set-Private-Key (pp, D, x): Given the system parameter pp, a partial private key D, and a secret value x, this algorithm outputs a private key sk.
- Set-Public-Key (pp, x): Given the system parameter pp and a secret value x, this algorithm outputs a public key pk.
- Set-Proxy-Key (pp, ID_P): Given the system parameter pp and a proxy identifier ID_P, this algorithm outputs a proxy public key pk_P and its secret value x_P.
- Enc(pp, pk, M, s): This algorithm outputs a ciphertext CT of a message M, with a designated number s, such that an equality test on CT must be performed with $s-1$ other ciphertexts with the same s.
- Dec(CT, sk): This algorithm outputs the message M or $\perp$.
- Aut(sk): This algorithm outputs a token tk that authorizes the tester to perform an equality test on the ciphertexts of users who own sk.
- Proxy-Aut (pp, sk, sk_P): This protocol outputs a token $\hat{tk}$ issued by the proxy who owns sk_P to the tester, enabling the latter to perform an equality test on the ciphertexts of the users who own sk.
- Test$(CT_1, \cdots, CT_t, \hat{tk}_1, \cdots, \hat{tkj}, tk_1, \cdots, tk_{t-j}$: Given t ciphertexts CT and t tokens, including tokens issued by users and the proxy, this algorithm checks whether $t = s_1 = s_2 = \cdots = s_t$, where s_i is the designated number of CT_i. If not, it outputs $\perp$ and aborts. Otherwise, it outputs 1, implying that the underlying messages of $CT_1, CT_2, \cdots, CT_t$ are equal, or 0, implying that the messages are not equal.

Correctness: We can say that a CLE-MET-PA scheme is correct if the following conditions hold.

(1) For any security parameter λ, any message $M \in \mathcal{M}$, and any number $s \in \mathbb{Z}_p$, we have

$$\Pr\left[\mathsf{Dec}(sk, CT) = M \middle| \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda) \\ D \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID) \\ x \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID) \\ sk \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D, x) \\ pk \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x) \\ (pk_P, x_P) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_P) \\ CT \leftarrow \mathsf{Enc}(pp, pk, M, s) \end{array}\right] = 1.$$

(2) For any security parameter λ, any message $M \in \mathcal{M}$, any number $t \in \mathbb{Z}_p$, and $i \in \{1, \cdots, t\}$, we have

$$\Pr\left[\begin{array}{l} \mathsf{Test}(CT_1, \cdots, CT_t, \hat{tk}_1, \cdots, \\ \hat{tk}_j, tk_1, \cdots, tk_{t-j}) = 1 \end{array} \middle| \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda) \\ D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \\ x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i) \\ sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_I, x_i) \\ pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i) \\ (pk_P, x_P) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i}) \\ CT_i \leftarrow \mathsf{Enc}(pp, pk_i, pk_{P_i}, M_i, t) \\ tk_i \leftarrow \mathsf{Aut}(sk_i) \\ \hat{tk}_i \leftarrow \mathsf{Proxy\text{-}Aut}(sk_i) \end{array}\right]$$

is overwhelming.

3.3. Security Models of CLE-MET-PA

We consider five types of adversaries in CLE-MET-PA. Type-I $\sim$ Type-IV exist in CL-PKEET and Type-V in PKE-MET. In CLE, users generate their own pairs of public and private keys. The absence of a certificate in the public key introduces a vulnerability to tampering. We define Type-I and Type-II adversaries as outsiders who possess the capability to replace legitimate public keys. In the context of CLE, where the KGC is not fully trusted, we define Type-III and Type-IV adversaries as curious KGCs who possess the system master secret key *msk*. For the discussion on the impact of trapdoors needed by the equality tests on system security, we define that Type-II and Type-IV adversaries can acquire all trapdoors in the system, including the trapdoor of the challenge ciphertext. Finally, considering the characteristics of PKE-MET, we define a Type-V adversary. The detailed descriptions are provided below:

- Type-I Adversary: This type of adversary can replace the public key of a user but cannot access the master key. Without the trapdoor of the challenge ciphertext, we define the IND-CPA security model regarding this type of adversary.
- Type-II Adversary: This type of adversary can replace the public key of a user but cannot access the master key. With the trapdoor of the challenge ciphertext, we define the OW-CPA security model regarding this type of adversary.
- Type-III Adversary: This type of adversary cannot replace the public key of a user but can access the master key. Without the trapdoor of the challenge ciphertext, we define the IND-CPA security model regarding this type of adversary.
- Type-IV Adversary: This type of adversary cannot replace the public key of a user but can access the master key. With the trapdoor of the challenge ciphertext, we define the OW-CPA security model regarding this type of adversary.

- Type-V Adversary: This type of adversary attempts to perform an equality test on t ciphertexts $CT_1, \cdots, CT_t$, where all the designated numbers of these ciphertexts are s_i, with $s_i > t$.

We define five games for these five types of adversaries.

Game 1: IND-CPA Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda);$$
$$D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \text{ for } 1 \le i \le N;$$
$$x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i)$$
$$sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_i, x_i)$$
$$pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i)$$
$$(pk_{P_i}, x_{P_i}) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i})$$
$$(s^*, pk^*, M_0^*, M_1^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}rep}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N);$$
$$CT^* \leftarrow \mathsf{Enc}(pk^*, M_b^*, s^*) \text{ for } b \in \{0,1\}, \text{random } s^*;$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}rep}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N).$$

In Game 1, $\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot)$, $\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot)$, $\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot)$, $\mathcal{O}^{\mathsf{pub\text{-}rep}}(\cdot)$, $\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot)$, $\mathcal{O}^{\mathsf{token}}(\cdot)$, and $\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)$, denote the partial key oracle, the private key oracle, the public key oracle, the public key replace oracle, the proxy key oracle, the token oracle, and the proxy token oracle, respectively. The adversary is not allowed to make private key query, token query, or proxy token query on pk^*. We define the advantage of adversary in winning this game as

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA,Type\text{-}I}}_{\mathsf{CLE\text{-}MET\text{-}PA}}(\lambda) = \Pr[b' = b] - 1/2.$$

Game 2: OW-CPA Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda);$$
$$D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \text{ for } 1 \le i \le N;$$
$$x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i)$$
$$sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_i, x_i)$$
$$pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i)$$
$$(pk_{P_i}, x_{P_i}) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i})$$
$$(s^*, pk^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}rep}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N);$$
$$CT^* \leftarrow \mathsf{Enc}(pk^*, M^*, s^*) \text{ for random } M^*, s^*;$$
$$M' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}rep}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N).$$

In Game 2, the adversary is restricted to making a private key query on pk^*. We define the advantage of the adversary in winning this game as

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA,Type\text{-}II}}_{\mathsf{CLE\text{-}MET\text{-}PA}}(\lambda) = \Pr[M' = M^*].$$

Game 3: IND-CPA Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda);$$
$$D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \text{ for } 1 \le i \le N;$$
$$x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i)$$
$$sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_i, x_i)$$
$$pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i)$$
$$(pk_{P_i}, x_{P_i}) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i})$$
$$(s^*, pk^*, M_0^*, M_1^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{msk}}(\cdot),\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N);$$
$$CT^* \leftarrow \mathsf{Enc}(pk^*, M_b^*, s^*) \text{ for } b \in \{0,1\}, \text{random } s^*;$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{msk}}(\cdot),\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N).$$

In Game 3, the adversary can access the master secret key oracle $\mathcal{O}^{\mathsf{msk}}(\cdot)$, but can no longer access the public key replace oracle. The adversary cannot make a private key, token

query, or proxy token query on pk^*. We define the advantage of the adversary in winning this game as

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA,Type\text{-}III}}_{\mathsf{CLE\text{-}MET\text{-}PA}}(\lambda) = \Pr[b' = b] - 1/2.$$

Game 4: OW-CPA Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda);$$
$$D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \text{ for } 1 \le i \le N;$$
$$x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i)$$
$$sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_i, x_i)$$
$$pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i)$$
$$(pk_{P_i}, x_{P_i}) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i})$$
$$(s^*, pk^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{msk}}(\cdot),\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N);$$
$$CT^* \leftarrow \mathsf{Enc}(pk^*, M^*, s^*) \text{ for random } M^*, s^*;$$
$$M' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{msk}}(\cdot),\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N).$$

In Game 4, the adversary can access the master secret key oracle $\mathcal{O}^{\mathsf{msk}}(\cdot)$, but can no longer access the public key replace oracle. The adversary cannot make a private key query on pk^*. We define the advantage of the adversary in winning this game as

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA,Type\text{-}IV}}_{\mathsf{CLE\text{-}MET\text{-}PA}}(\lambda) = \Pr[M' = M^*].$$

Game 5: Number Game

$$pp \leftarrow \mathsf{Setup}(1^\lambda);$$
$$D_i \leftarrow \mathsf{Partial\text{-}Private\text{-}Key\text{-}Extract}(pp, msk, ID_i) \text{ for } 1 \le i \le N;$$
$$x_i \leftarrow \mathsf{Set\text{-}Secret\text{-}Value}(pp, ID_i)$$
$$sk_i \leftarrow \mathsf{Set\text{-}Private\text{-}Key}(pp, D_i, x_i)$$
$$pk_i \leftarrow \mathsf{Set\text{-}Public\text{-}Key}(pp, x_i)$$
$$(pk_{P_i}, x_{P_i}) \leftarrow \mathsf{Set\text{-}Proxy\text{-}Key}(pp, ID_{P_i})$$
$$(s^*, t^*, pk_1^*, \cdots, pk_t^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N), t^* < s^*;$$
$$(CT_1^*, \cdots, CT_j^*, \cdots, CT_{t^*}^*) \leftarrow \mathsf{Enc}((M_0^*, pk_1^*, s^*), \cdots, (M_0^*, pk_{j-1}^*, s^*), (M_b^*, pk_j^*, s^*), (M_0^*, pk_{j+1}^*, s^*), \cdots, (M_0^*, pk_t^*, s^*))$$
$$\text{for random messages } M_0^*, M_1^* \text{ and number } b \in \{0,1\};$$
$$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{par\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{priv\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{pub\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}key}}(\cdot),\mathcal{O}^{\mathsf{token}}(\cdot),\mathcal{O}^{\mathsf{p\text{-}token}}(\cdot)}(\{pk_i, pk_{P_i}\}_{i=1}^N).$$

In Game 5, the adversary cannot make private key queries on $pk_1^*, \cdots, pk_{t^*}^*$. We define the advantage of the adversary in winning this game as

$$\mathsf{Adv}^{\mathsf{Number,Type\text{-}V}}_{\mathsf{CLE\text{-}MET\text{-}PA}}(\lambda) = \Pr[b' = b] - 1/2.$$

4. The Proposed CLE-MET-PA Scheme

In certificateless encryption supporting multi-ciphertext equality test with proxy-assisted authorization (CLE-MET-PA), each ciphertext CT can designate a number s. With this designation, an authorized cloud server is enabled to perform an equality test on s ciphertexts, but only if all the designated numbers of these s ciphertexts are equal to s.

4.1. Our Construction

$\mathsf{Setup}(1^\lambda)$: Taking as input a security parameter λ, the setup algorithm generates a bilinear groups ensemble $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and five cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_2$, $H_2 : \{0,1\}^* \to \mathbb{G}_2$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$, $H_4 : \{0,1\}^* \to \{0,1\}^{2l}$, $H_5 : \{0,1\}^* \to \{0,1\}^\lambda$, where l is the bit length of a group element in $\mathbb{Z}_p$. Pick a random $\alpha \in \mathbb{Z}_p^*$, $\bar{g} = g_1^\alpha$. It sets the system parameter as

$$pp = \{\mathcal{G}, g_1, g_2, \bar{g}, H_1, H_2, H_3, H_4, H_5\},$$

and the master secret key

$$msk = \alpha.$$

Partial private key extract (pp, msk, ID): Taking as input the system parameter pp, the master key msk, and the identifier ID. The key generation algorithm computes and outputs the partial private key pair

$$D = (D_1, D_2) = (H_1(ID)^{\alpha}, H_2(ID)^{\alpha}).$$

Set-Secret-Value (pp, ID): Taking as input the system parameter pp and an identifier ID. The algorithm picks a value $x \in \mathbb{Z}_q^*$ at random. A proxy secret value corresponding to ID_P is denoted as x_P.

Set-Private-Key (pp, D, x): Taking as input the system parameter pp, a partial private key pair and an secret value x. The algorithm computes the private key pair sk, where

$$sk = (sk_1, sk_2) = (D_1^x, D_2^x) = (H_1(ID)^{\alpha x}, H_2(ID)^{\alpha x}).$$

Set-Public-Key (pp, x): Taking as input the system parameter pp and a secret value x, the algorithm outputs the public key pair

$$pk = (X = \bar{g}^x, Y = g_2^x, Z = g_1^x) = (g_1^{\alpha x}, g_2^x, g_1^x).$$

Set-Proxy-Key (pp, ID_P): The proxy can be any user. A proxy generates the secret value and public key by running Set-Secret-Value (pp, ID_P) and Set-Public-Key (pp, x_P). Output the secret key x_P, and the public key pair

$$pk_P = (X_P = \bar{g}^{x_P}, Y_P = g_2^{x_P}, Z_P = g_1^{x_P}) = (g_1^{\alpha x_P}, g_2^{x_P}, g_1^{x_P}).$$

Enc(pp, pk, M, s): Taking as input the system parameter pp, a user public key pk, check whether $e(X, g_2) = e(\bar{g}, Y)$ and $e(Z, g_2) = e(g_1, Y)$ holds; if not, output $\perp$ and abort. Then, taking as input a message $M \in \mathbb{Z}_p$, and a number $s \in \mathbb{Z}_p$, the encryption algorithm iteratively computes

$$\begin{aligned} &f_0 = H_3(M||s), \quad f_1 = H_3(M||s||f_0), \quad \cdots, \\ &f_{s-1} = H_3(M||s||f_0||\cdots||f_{s-2}) \end{aligned}$$

as illustrated in Figure 3 with $i = s$. Let

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{s-1} x^{s-1}.$$

It randomly chooses $A, r_1, r_2 \in \mathbb{Z}_p$, and outputs the ciphertext $CT = (s, C_1, C_2, C_3, C_4, C_5, C_6)$ as

$$\begin{aligned} &C_1 = g_1^{r_1}, \quad C_2 = H_4(e(X, H_1(ID))^{r_1}) \oplus (M||r_1), \\ &C_3 = g_1^{r_2}, \quad C_4 = Z^{r_2}, \\ &C_5 = H_4(e(X, H_2(ID))^{r_2}) \oplus (A||f(A)), \\ &C_6 = H_5(s||C_1||C_2||C_3||C_4||C_5||e(X, H_2(ID))^{r_2}||f_0||f_1||\cdots||f_{s-1}). \end{aligned}$$

Note that the equality of this ciphertext CT can only be performed as an equality test with other $s-1$ ciphertexts in which all the designated numbers are s.

Dec(CT, sk): Taking as input a ciphertext $CT = (s, C_1, C_2, C_3, C_4, C_5, C_6)$ and a secret key $sk = (H_1(ID)^{\alpha x}, H_2(ID)^{\alpha x})$, the decrypt algorithm computes

$$M'||r_1' = C_2 \oplus H_4(e(C_1, sk_1))$$

It then computes

$$f'_0 = H_3(M'||s), \quad f'_1 = H_3(M'||s||f'_0), \quad \cdots,$$
$$f'_{s-1} = H_3(M'||s||f'_0||\cdots||f'_{s-2})$$

and checks whether the following equations hold or not

$$C_1 = g_1^{r'_1},$$
$$f'(A') = f'_0 + f'_1 A' + \cdots + f'_{s-1} A'^{s-1},$$
$$C_6 = H_5(s||C_1||C_2||C_3||C_4||C_5||e(C_3, sk_2)||f'_0||f'_1||\cdots||f'_{s-1}),$$

where $A'||f'(A') = C_5 \oplus H_4(e(C_3, sk_2))$. If all the equations hold, it returns

$$M = M'.$$

Otherwise, it returns $\perp$.

$\mathsf{Aut}(sk)$: Taking as input a secret key $sk = (sk_1, sk_2) = (H_1(ID)^{\alpha x}, H_2(ID)^{\alpha x})$, the authorization algorithm returns the token as

$$tk = sk_2 = H_2(ID)^{\alpha x}.$$

$\mathsf{Proxy\text{-}Aut}\ (pp, sk, sk_P)$: Following the algorithm depicted in Table 1, the user receives and publishes $H_2(ID)^{x_P}$ as the proxy information PI,

$$PI = H_2(ID)^{x_P},$$

and the algorithm returns the proxy token as

$$\hat{tk} = H_2(ID)^{\alpha x + x \cdot x_P}.$$

Table 1. Key exchange protocol between the user and proxy.

Key Exchange Protocol
The user checks whether $e(X_P, g_2) = e(\bar{g}, Y_P)$ and $e(Z_P, g_2) = e(g_1, Y_P)$ holds, if not, output $\perp$ and abort.
User $\xleftarrow{H_2(ID)^{x_P}}$ Proxy
The user checks whether $e(Z_P, H_2(ID)) = e(g_1, H_2(ID)^{x_P})$ holds, if not, abort and output $\perp$. Otherwise, the user outputs proxy information $PI = H_2(ID)^{x_P}$, computes $\hat{tk} = sk_2 \cdot (PI)^x = H_2(ID)^{\alpha x + x \cdot x_P}$.
User $\xrightarrow{\hat{tk}}$ Proxy
The proxy receives the proxy token $\hat{tk}$ without revealing x_P and knowing part of sk.

$\mathsf{Test}(CT_1, \cdots, CT_t, \hat{tk}_1, \cdots, \hat{tk}_j, tk_1, \cdots, tk_{t-j})$: Taking as input t ciphertexts $CT_1, \cdots, CT_t$ where $CT_i = (s_i, C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6})$ and t corresponding tokens including the proxy token and user token: $\hat{tk}_1, \cdots, \hat{tk}_j, tk_1, \cdots, tk_{t-j}$, the test algorithm aborts if the equation $s_1 = \cdots = s_t = t$ does not hold. Otherwise, for each $i \in \{1, \cdots, j\}$:

- With the token authorized by the user, it computes

$$A_i||f_i(A_i) = C_{i,5} \oplus H_4(e(C_{i,3}, tk_i)),$$

- With the proxy token authorized by the proxy, it computes

$$A_i||f_i(A_i) = C_{i,5} \oplus H_4\Big(e(C_{i,3}, \hat{tk}_i)/e(C_{i,4}, PI_i)\Big),$$

where

$$f_i(A_i) = f_{i,0} + f_{i,1}A_i + f_{i,2}A_i^2 + \cdots + f_{i,t-1}A_i^{t-1}.$$

With A_i and $f_i(A_i)$ for $i \in \{1, \cdots, t\}$, we have an equation set as

$$\begin{cases} f_1(A_1) = f_{1,0} + f_{1,1}A_1 + \cdots + f_{1,t-1}A_1^{t-1} \\ f_2(A_2) = f_{2,0} + f_{2,1}A_2 + \cdots + f_{2,t-1}A_2^{t-1} \\ \quad\quad \vdots \\ f_t(A_t) = f_{t,0} + f_{t,1}A_t + \cdots + f_{t,t-1}A_t^{t-1} \end{cases}.$$

Implicitly setting $f_{i,k} = f_{j,k}$ for $i, j \in \{1, 2, \cdots, t\}$ and $k \in \{0, 1, \cdots, t-1\}$, it solves the equation set and obtains a unique solution $f_{i,0}, f_{i,1}, \cdots, f_{i,t-1}$. It checks whether the following equations hold or not for each $i \in \{1, 2, \cdots, j\}$.

$$C_{i,6} = H_5(t||C_{i,1}|| \cdots ||C_{i,5}||e(C_{i,3}, \hat{tk}_i)/e(C_{i,4}, PI_i)||f_{i,0}|| \cdots ||f_{i,t-1}),$$

and for each $i \in \{1, 2, \cdots, t-j\}$.

$$C_{i,6} = H_5(t||C_{i,1}|| \cdots ||C_{i,5}||e(C_{i,3}, tk_i)||f_{i,0}|| \cdots ||f_{i,t-1}).$$

- If all the equations hold, it returns 1, implying that $M_1 = M_2 = \cdots = M_t$.
- Otherwise, it returns 0, implying that the equation $M_1 = M_2 = \cdots = M_t$ does not hold.

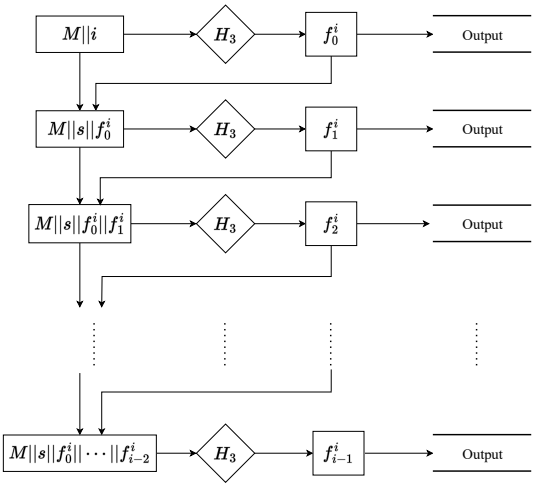

Figure 3. Iteration computation of $\{f_j^n\}_{0 \le j \le n-1}$.

4.2. Correctness of CLE-MET-PA

We analyze the correctness of the proposed CLE-MET-PA construction as below.

(1) In the decryption algorithm, denoted by Dec, the decryption process computes the following:

$$\begin{aligned} & C_2 \oplus H_4(sk_1, C_1) \\ =& H_4(e(X, H_1(ID))^{r_1}) \oplus (M||r_1) \oplus H_4(e(sk_1, C_1)) \\ =& H_4(e(g_1^{\alpha x}, H_1(ID))^{r_1}) \oplus (M||r_1) \oplus H_4\big(e(H_1(ID)^{\alpha x}, g_1^{r_1})\big) \\ =& M||r_1 \end{aligned}$$

The output M and r_1 satisfy the three listed equations. Subsequently, the decryption algorithm will recover the message M, implying that

$$\Pr\left[\mathsf{Dec}(sk, CT) = M\right] = 1.$$

(2) In the test algorithm, denoted by Test, after implicitly setting $f_{i,k} = f_{j,k}$ for $i, j \in 1, 2, \cdots, t$ and $k \in 0, 1, \cdots, t-1$, we obtain an equation set comprising t equations for t unknown variables. We can then represent the coefficients of this equation set as the following Vandermonde matrix:

$$V = \begin{bmatrix} 1 & A_1 & A_1^2 & \cdots & A_1^{t-1} \\ 1 & A_2 & A_2^2 & \cdots & A_2^{t-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & A_t & A_t^2 & \cdots & A_t^{t-1} \end{bmatrix}.$$

Thanks to the properties of the Vandermonde matrix, the equation set possesses a unique solution when the determinant of this matrix, denoted by $\det(V)$, satisfies $\det(V) \neq 0$. This determinant is defined as:

$$\det(V) = \prod_{1 \leq i < j \leq t} (A_i - A_j).$$

Since the values of A_i for $i \in 1, \cdots, t$ are randomly chosen from the set $\mathbb{Z}_p$, we can conclude that

$$\det(V) \neq 0$$

holds with overwhelming probability, specifically $\frac{p(p-1)\cdots(p-t+1)}{p^t}$.

Consequently, we obtain a unique solution for the variables $f_{1,0}, f_{1,1}, \cdots, f_{1,t-1}$. Given that $M_1 = \cdots = M_t$, it follows that $f_{i,k} = f_{j,k}$ for $i, j \in 1, 2, \cdots, t$ and $k \in 0, 1, \cdots, t-1$. Thus, the unique solution will satisfy the following equations for $1 \leq i \leq j \leq t$.

$$C_{i,6} = H_5(s||C_{i,1}|| \cdots ||C_{i,5}||e(C_{i,3}, \hat{tk}_i)/e(C_{i,4}, PI_i)||f_{1,0}|| \cdots ||f_{1,j})$$
$$C_{i,6} = H_5(s||C_{i,1}|| \cdots ||C_{i,5}||e(C_{i,3}, tk_i)||f_{1,0}|| \cdots ||f_{1,t-j+1})$$

Then, we have

$$\mathsf{Test}(CT_1, \cdots, CT, \hat{tk}_1, \cdots, \hat{tk}_j, tk_1, \cdots, tk_{t-j}) = 1$$

will hold with overwhelming probability.

(3) Similarly to the previous step (2), we can obtain a unique solution from the equation set. However, since the equation $M_1 = \cdots = M_t$ does not hold, without a loss of generality, assume that $M_1 \neq M_2 = M_3 = \cdots = M_t$. In this scenario, we then have $f_{1,k} \neq f_{i,k} = f_{j,k}$ for $i, j \in 2, \cdots, t, k \in 0, \cdots, t-1$. Nevertheless, it is evident that the solution cannot simultaneously satisfy the two following equations.

$$C_{1,6} = H_5(t||C_{1,1}|| \cdots ||C_{1,4}||e(C_{1,3}, \hat{tk}_1)/e(C_{1,4}, PI_1)||f'_{1,0}|| \cdots ||f'_{1,j})$$
$$C_{2,6} = H_5(t||C_{2,1}|| \cdots ||C_{2,4}||e(C_{2,3}, \hat{tk}_2)/e(C_{1,4}, PI_2)||f'_{1,0}|| \cdots ||f'_{1,j})$$

Therefore, the test algorithm will output 0. We have

$$\mathsf{Test}(CT_1, \cdots, CT, \hat{tk}_1, \cdots, \hat{tk}_j, tk_1, \cdots, tk_{t-j}) = 0$$

will hold with overwhelming probability.

5. Security Proof

Our CLE-MET-PA construction is secure due to the hardness of the BDH assumption.

Theorem 1. *For any PPT Type-I adversary, our CLE-MET-PA scheme is IND-CPA-secure based on the BDH assumption in the random oracle model.*

Proof of Theorem 1. Assume there exists an adversary $\mathcal{A}_1$ who can break the IND-CPA security of our scheme with a non-negligible advantage ϵ, we can construct a simulator $\mathcal{B}$ to break the BDH assumption. Given an instance as $(\mathcal{G}, g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$, $\mathcal{B}$ is to compute $e(g_1, g_2)^{abc}$ by running $\mathcal{A}_1$ as a subroutine. $\mathcal{B}$ and $\mathcal{A}_1$ play the following game.

1. *Setup:* $\mathcal{B}$ randomly picks a cryptographic hash function $H_3 : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ and sets the public parameter $pp = \{\mathcal{G}, g_1, g_2, \bar{g}, H_1, H_2, H_3, H_4, H_5\}$, where $\bar{g} = g_1^a$ and H_1, H_2, H_4, H_5 are random oracles. pp is sent to $\mathcal{A}_1$. Lists $L_{H_1}, L_{H_2}, L_{H_4}, L_{H_5}, and L_s$ are initially empty. Assume that $\mathcal{A}_1$ can make $q_{H_1}, q_{H_2}, q_{H_4}, q_{H_5}$ hash queries to $H_1, H_2, H_4, and H_5$, respectively.
2. *Phase 1:* Assume there are n users with identities $ID_1, \cdots, ID_n$ in the system. $\mathcal{B}$ randomly chooses $i^* \in [1, n]$ and performs the following steps.
   - *H_1-query (ID_i):* For i-th query ID_i, $\mathcal{B}$ searches L_{H_1}. If there exists the related item of ID_i as $(u_i, ID_i, H_1(ID_i))$, $\mathcal{B}$ returns $H_1(ID_i)$ to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly chooses u_i and sets
   $$H_1(ID_i) = \begin{cases} g_2^{u_i}, & if\ i \neq i^*, \\ g_2^{bu_i}, & if\ i = i^*. \end{cases}$$
   $\mathcal{B}$ stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1} and returns $H_1(ID_i)$ to $\mathcal{A}_1$.
   - *H_2-query (ID_i):* For the i-th query ID_i, $\mathcal{B}$ searches L_{H_2}. If there exists the related item of ID_i as $(v_i, ID_i, H_2(ID_i))$, $\mathcal{B}$ returns $H_2(ID_i)$ to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly chooses v_i and sets
   $$H_2(ID_i) = \begin{cases} g_2^{v_i}, & if\ i \neq i^*, \\ g_2^{bv_i}, & if\ i = i^*. \end{cases}$$
   $\mathcal{B}$ stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2} and returns $H_2(ID_i)$ to $\mathcal{A}_1$.
   - *H_4-query (Q_i):* For i-th query Q_i, $\mathcal{B}$ randomly picks $\eta_i \in \{0,1\}^{2l}$, sets $H_4(Q_i) = \eta_i$, stores a new item $(Q_i, H_4(Q_i) = \eta_i)$ into L_{H_4}, and then returns $H_4(Q_i)$ to $\mathcal{A}_1$.
   - *H_5-query (W_i):* For the i-th query W_i, $\mathcal{B}$ randomly picks $\sigma_i \in \{0,1\}^\lambda$, sets $H_5(W_i) = \sigma_i$, stores a new item $(W_i, H_5(W_i) = \sigma_i)$ into L_{H_5}, and then returns $H_5(W_i)$ to $\mathcal{A}_1$.
   - *Partial private key query (ID_i):* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets
   $$H_1(ID_i) = g_2^{u_i},$$
   and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets
   $$H_2(ID_i) = g_2^{v_i},$$
   and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ then computes
   $$D_i = (D_{i,1}, D_{i,2}) = (H_1^\alpha(ID), H_2^\alpha(ID)) = \left(g_2^{au_i}, g_2^{av_i}\right),$$
   which can be computed with a known g_2^a, u_i, v_i. $\mathcal{B}$ returns D_i to $\mathcal{A}_1$.
   - *Secret key query (ID_i):* For the i-th queried identity ID_i, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$, stores $(ID_i, x_i, -, -)$ into L_s, and returns x_i to $\mathcal{A}_1$.

- *Private key query (ID_i):* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{ax_iu_i}, g_2^{ax_iv_i}\right),$$

which can be computed with known g_2^a, x_i, u_i, v_i. It then stores $(ID_i, x_i, sk_i, -)$ into L_s and returns sk_i to $\mathcal{A}_1$.
- *Public key query (ID_i):* For i-th queried identity ID_i, $\mathcal{B}$ searches L_s, and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{ax_i}, g_2^{x_i}, g_1^{x_i}\right),$$

which can be computed with known g_1, g_1^a, g_2, x_i. It then stores $(ID_i, x_i, -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_1$.
- *Proxy key query (ID_{P_i}):* For a proxy with the related item in L_s as (ID_{P_i}, x_{P_i}), $\mathcal{B}$ performs as *Public key query* step and obtains

$$pk_{P_i} = (X_{P_i}, Y_{P_i}, Z_{P_i}) = \left(g_1^{ax_{P_i}}, g_2^{x_{P_i}}, g_1^{x_{P_i}}\right).$$

which can be computed with known g_1, g_1^a, g_2, x_{P_i}. $\mathcal{B}$ then returns pk_{P_i} and sk_{P_i} to $\mathcal{A}_1$.
- *Public key replace query (ID_i, pk'):* $\mathcal{B}$ changes the public key pk_i corresponding to ID_i to $pk' = (X', Y', Z')$ while receiving (ID_i, pk') if $e(X_i', g_2) = e(\bar{g}_1, Y_i')$ and $e(Z_i', g_2) = e(g_1, Y_i')$ holds.
- *Token query (ID_i):* For a queried ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches L_s to find the related item of ID_i as (ID_i, x_i, sk_i), where $sk_i = (sk_1, sk_2)$. If the related item does not exist, $\mathcal{B}$ performs as follows. If ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{ax_iu_i}, g_2^{ax_iv_i}\right),$$

which can be computed with known g_2^a, x_i, u_i, v_i. It then stores (ID_i, x_i, sk_i) into L_s. $\mathcal{B}$ returns sk_2 to $\mathcal{A}_1$.

- *Proxy token query (ID_i, ID_{P_i}):* For a queried ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ performs as Aut step except that $\mathcal{B}$ computes $PI_i = H_2(ID_i)^{x_{P_i}} = g_2^{av_i x_i x_{P_i}}$, and the proxy token

$$\hat{tk}_i = H_2(ID)^{\alpha x + x \cdot x_P} = g_2^{av_i x_i + x_i x_{P_i}}$$

with known $g_2, g_2^a, v_i, x_i, x_{P_i}$. $\mathcal{B}$ returns $\hat{tk}_i$ to $\mathcal{A}_1$.

3. *Challenge:* $\mathcal{A}_1$ sends $(s^*, ID^*, M_0^*, M_1^*)$ to $\mathcal{B}$, where s^* represents the designated challenge number, ID^* stands for the challenge identity, and two plaintexts M_0^* and M_1^* are selected from $\{0,1\}^\lambda$ with equal lengths. If $ID^* \neq ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, it randomly picks $w_1, w_2 \in \mathbb{Z}_p, R_1, R_2 \in \{0,1\}^{2l}, R_3 \in \{0,1\}^\lambda$ and implicitly sets $r_1 = w_1 c, r_2 = w_2 c$. It then computes the challenge ciphertext as

$$\begin{aligned}
C_1 &= g_1^{r_1} = g_1^{cw_1}, \\
C_2 &= R_1, \\
C_3 &= g_1^{r_2} = g_1^{cw_2}, \\
C_4 &= g_1^{x_{i^*} r_2} = g_1^{c x_{i^*} w_2}, \\
C_5 &= R_2, \\
C_6 &= R_3.
\end{aligned}$$

4. *Phase 2:* $\mathcal{B}$ interacts with $\mathcal{A}_1$ as *Phase 1* with the limitation that ID_{i^*} cannot be queried in *partial private key query*, *secret key query*, *private key query*, *token query*, and *proxy token query*.
5. Guess: $\mathcal{A}_1$ outputs its guess bit $\rho' \in \{0,1\}$.
6. *Solve:* $\mathcal{B}$ randomly chooses an item $(Q_i, H_4(Q_i) = \eta_i)$ from L_{H_4} and sets

$$e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*} u_{i^*} w_1}}$$

as the solution to the BDH instance.

7. *Analysis:* To successfully perform the reduction, the simulation should be indistinguishable from the real attack from the point of view of the adversary. As we can see, if the adversary chooses ID_{i^*} as the challenge identity, the simulation will not abort, which means that the simulation is indistinguishable from the real attack. The corresponding probability is $1/n$. Upon the case that the simulation is indistinguishable to the adversary, we have the following analysis. Since the adversary is assumed to break the security with the advantage ϵ, we have that it issues the hash query $e(X_{i^*}, H_1(ID_{i^*}))^{r_1} = e(g_1^{ax_{i^*}}, g_2^{bu_{i^*}})^{cw_1} = e(g_1, g_2)^{abc \cdot x_{i^*} u_{i^*} w_1}$ with probability ϵ. Thus, $\mathcal{B}$ can finally obtain the true solution to the given BDH instance as $e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*} u_{i^*} w_1}}$ with a probability of $\frac{\epsilon}{q_{H_4}}$. In conclusion, $\mathcal{B}$ can successfully break the BDH assumption with the probability of

$$\Pr[\mathcal{B} \textit{ chooses the correct hash query} | \textit{the simulation is indistinguishable}] = \frac{\epsilon}{n \cdot q_{H_4}}.$$

□

Theorem 2. *For any PPT Type-II adversary, our CLE-MET-PA scheme is OW-CPA secure based on the BDH assumption in the random oracle model.*

Proof of Theorem 2. Assume there exists an adversary $\mathcal{A}_2$ who can break the OW-CPA security of our scheme with non-negligible advantage ϵ; then, we can construct a simulator

$\mathcal{B}$ to break the BDH assumption. Given an instance as $(\mathcal{G}, g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$, $\mathcal{B}$ is to compute $e(g_1, g_2)^{abc}$ by running $\mathcal{A}_2$ as a subroutine. $\mathcal{B}$ and $\mathcal{A}_2$ play the following game.

1. *Setup:* $\mathcal{B}$ randomly picks a cryptographic hash function $H_3 : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ and sets the public parameter $pp = \{\mathcal{G}, g_1, g_2, \bar{g}, H_1, H_2, H_3, H_4, H_5\}$, where $\bar{g} = g_1^a$ and H_1, H_2, H_4, H_5 are random oracles. pp is sent to $\mathcal{A}_2$. Lists $L_{H_1}, L_{H_2}, L_{H_4}, L_{H_5}, L_s$ are initially empty. Assume that $\mathcal{A}_2$ can make $q_{H_1}, q_{H_2}, q_{H_4}, q_{H_5}$ hash queries to H_1, H_2, H_4, H_5, respectively.
2. *Phase 1:* Assume there are n users with identities $ID_1, \cdots, ID_n$ in the system. $\mathcal{B}$ randomly chooses $i^* \in [1, n]$ and performs the following steps.
   - *H_1-query (ID_i):* For i-th query ID_i, $\mathcal{B}$ searches L_{H_1}. If there exists the related item of ID_i as $(u_i, ID_i, H_1(ID_i))$, $\mathcal{B}$ returns $H_1(ID_i)$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ randomly chooses u_i and sets
   $$H_1(ID_i) = \begin{cases} g_2^{u_i}, & if\ i \neq i^*, \\ g_2^{bu_i}, & if\ i = i^*. \end{cases}$$
   $\mathcal{B}$ stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1} and returns $H_1(ID_i)$ to $\mathcal{A}_2$.
   - *H_2-query (ID_i):* For i-th query ID_i, $\mathcal{B}$ searches L_{H_2}. If there exists the related item of ID_i as $(v_i, ID_i, H_2(ID_i))$, $\mathcal{B}$ returns $H_2(ID_i)$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ randomly chooses v_i and sets
   $$H_2(ID_i) = g_2^{v_i}$$
   $\mathcal{B}$ stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2} and returns $H_2(ID_i)$ to $\mathcal{A}_2$.
   - *H_4-query (Q_i):* For the i-th query Q_i, $\mathcal{B}$ picks $\eta_i \in \{0,1\}^{2l}$ randomly, sets $H_4(Q_i) = \eta_i$, stores a new item $(Q_i, H_4(Q_i) = \eta_i)$ into L_{H_4}, and then returns $H_4(Q_i)$ to $\mathcal{A}_2$.
   - *H_5-query (W_i):* For the i-th query W_i, $\mathcal{B}$ randomly picks $\sigma_i \in \{0,1\}^\lambda$, sets $H_5(W_i) = \sigma_i$, stores a new item $(W_i, H_5(W_i) = \sigma_i)$ into L_{H_5}, and then returns $H_5(W_i)$ to $\mathcal{A}_2$.
   - *Partial private key query (ID_i):* For the i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets
   $$H_1(ID_i) = g_2^{u_i},$$
   and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets
   $$H_2(ID_i) = g_2^{v_i},$$
   and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ then computes
   $$D_i = (D_{i,1}, D_{i,2}) = (H_1^\alpha(ID), H_2^\alpha(ID)) = \left(g_2^{au_i}, g_2^{av_i}\right),$$
   which can be computed with known g_2^a, u_i, v_i. $\mathcal{B}$ returns D_i to $\mathcal{A}_2$.
   - *Secret key query (ID_i):* For the i-th queried identity ID_i, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$, stores $(ID_i, x_i, -, -)$ into L_s, and returns x_i to $\mathcal{A}_2$.
   - *Private key query (ID_i):* For the i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets
   $$H_1(ID_i) = g_2^{u_i},$$
   and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets
   $$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{ax_iu_i}, g_2^{ax_iv_i}\right),$$

which can be computed with known g_2^a, x_i, u_i, v_i. It then stores $(ID_i, x_i, sk_i, -)$ into L_s and returns sk_i to $\mathcal{A}_2$.

- *Public key query (ID_i):* For the i-th queried identity ID_i, $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{ax_i}, g_2^{x_i}, g_1^{x_i}\right),$$

which can be computed with a known g_1, g_1^a, g_2, x_i. It then stores $(ID_i, x_i, -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_2$.

- *Proxy key query (ID_{P_i}):* For a proxy with the related item in L_s as (ID_{P_i}, x_{P_i}), $\mathcal{B}$ performs as the *public key query* step and obtains

$$pk_{P_i} = (X_{P_i}, Y_{P_i}, Z_{P_i}) = \left(g_1^{ax_{P_i}}, g_2^{x_{P_i}}, g_1^{x_{P_i}}\right).$$

which can be computed with known g_1, g_1^a, g_2, x_{P_i}. $\mathcal{B}$ then returns pk_{P_i} and sk_{P_i} to $\mathcal{A}_2$.

- *Public key replace query (ID_i, pk'):* $\mathcal{B}$ changes the public key pk_i corresponding to ID_i to $pk' = (X', Y', Z')$ while receiving (ID_i, pk') if $e(X_i', g_2) = e(\bar{g}_1, Y_i')$ and $e(Z_i', g_2) = e(g_1, Y_i')$ holds.
- *Token query (ID_i):* For a queried ID_i, $\mathcal{B}$ searches L_s to find the related item of ID_i as (ID_i, x_i, sk_i), where $sk_i = (sk_1, sk_2)$. If the related item does not exist, $\mathcal{B}$ performs as follows. If ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{ax_iu_i}, g_2^{ax_iv_i}\right),$$

which can be computed with known g_2^a, x, u_i, v_i. It then stores (ID_i, x_i, sk_i) into L_s. $\mathcal{B}$ returns sk_2 to $\mathcal{A}_2$.

- *Proxy token query (ID_i, ID_{P_i}):* For a queried ID_i, $\mathcal{B}$ performs as Aut step except that $\mathcal{B}$ computes $PI_i = H_2(ID_i)^{x_{P_i}} = g_2^{av_ix_ix_{P_i}}$, and the proxy token

$$\hat{tk}_i = H_2(ID)^{\alpha x + x \cdot x_P} = g_2^{av_ix_i + x_ix_{P_i}}$$

with known $g_2, g_2^a, v_i, x_i, x_{P_i}$. $\mathcal{B}$ returns $\hat{tk}_i$ to $\mathcal{A}_2$.

3. *Challenge:* $\mathcal{A}_2$ sends (s^*, ID^*) to $\mathcal{B}$, where s^* represents the designated challenge number, and ID^* stands for the challenge identity. If $ID^* \neq ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, it chooses to randomly pick $M^* \in \{0,1\}^{\lambda}, w_1, w_2 \in \mathbb{Z}_p, R_1 \in \{0,1\}^{2l}$, and

implicitly sets $r_1 = w_1 c, r_2 = w_2 c$. Taking as input M^*, s^*, it then iteratively computes $(f_0, f_1, \cdots, f_{s^*-1})$ and $f(x)$ same as $\mathsf{Enc}(pp, pk, M, s)$ in Section 4.1. Then, it randomly picks $A \in \mathbb{Z}_p$, computes $f(A)$, then outputs the challenge ciphertext as follows:

$$
\begin{aligned}
&C_1 = g_1^{r_1} = g_1^{cw_1},\\
&C_2 = R_1,\\
&C_3 = g_1^{r_2} = g_1^{cw_2},\\
&C_4 = g_1^{x_{i^*} r_2} = g_1^{c x_{i^*} w_2},\\
&C_5 = H_4\Big(e(g_1^{cw_1}, g_2^{a v_{i^*} x_{i^*}})\Big) \oplus (A || f(A)),\\
&C_6 = H_5\Big(s^* || C_1^* || C_2^* || C_3^* || C_4^* || C_5^* || e(g_1^{cw_1}, g_2^{a v_{i^*} x_{i^*}}) || f_0 || f_1 || \cdots || f_{s^*-1}\Big).
\end{aligned}
$$

4. *Phase 2:* $\mathcal{B}$ interacts with $\mathcal{A}_2$ as *Phase 1* with the limitation that ID_{i^*} cannot be queried in *partial private key query*, *secret key query*, and *private key query*.
5. Guess: $\mathcal{A}_2$ outputs its guess $M' \in \{0,1\}^\lambda$.
6. *Solve:* $\mathcal{B}$ randomly chooses an item $(Q_i, H_4(Q_i) = \eta_i)$ from L_{H_4} and sets

$$e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*} u_{i^*} w_1}}$$

as the solution to the BDH instance.

7. *Analysis:* To successfully perform the reduction, the simulation should be indistinguishable from the real attack from the point of view of the adversary. As we can see, if the adversary chooses ID_{i^*} as the challenge identity, the simulation will not abort, which means the simulation is indistinguishable from the real attack. The corresponding probability is $1/n$. Upon the case that the simulation is indistinguishable to the adversary, we have the following analysis. Since the adversary is assumed to break the security with advantage ϵ, we have that it issues the hash query $e(X_{i^*}, H_1(ID_{i^*}))^{r_1} = e(g_1^{a x_{i^*}}, g_2^{b u_{i^*}})^{cw_1} = e(g_1, g_2)^{abc \cdot x_{i^*} u_{i^*} w_1}$ with probability ϵ. Thus, $\mathcal{B}$ finally can obtain the true solution to the given BDH instance as $e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*} u_{i^*} w_1}}$ with probability $\frac{\epsilon}{q_{H_4}}$. In conclusion, $\mathcal{B}$ can successfully break the BDH assumption with probability

$$\Pr[\mathcal{B} \textit{ chooses the correct hash query} | \textit{the simulation is indistinguishable}] = \frac{\epsilon}{n \cdot q_{H_4}}.$$

□

Theorem 3. *For any PPT Type-III adversary, our CLE-MET-PA scheme is IND-CPA secure based on the BDH assumption in the random oracle model.*

Proof of Theorem 3. Assume that there exists an adversary $\mathcal{A}_3$ who can break the IND-CPA security of our scheme with the non-negligible advantage ϵ, we can construct a simulator $\mathcal{B}$ to break the BDH assumption. Given an instance as $(\mathcal{G}, g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$, $\mathcal{B}$ is to compute $e(g_1, g_2)^{abc}$ by running $\mathcal{A}_3$ as a subroutine. $\mathcal{B}$ and $\mathcal{A}_3$ play the following game.

1. *Setup:* $\mathcal{B}$ randomly picks a cryptographic hash function $H_3 : \{0,1\}^* \to \{0,1\}^\lambda$, randomly picks $\alpha \in \mathbb{Z}_p^*$, and sets the public parameter pp. $pp = \{\mathcal{G}, g_1, g_2, \bar{g}, H_1, H_2, H_3, H_4, H_5\}$, where $\bar{g} = g_1^\alpha$, H_1, H_2, H_4, H_5 are random oracles. pp is sent to $\mathcal{A}_3$. Lists $L_{H_1}, L_{H_2}, L_{H_4}, L_{H_5}, L_s$ are initially empty. Assume that $\mathcal{A}_3$ can make $q_{H_1}, q_{H_2}, q_{H_4}, q_{H_5}$ hash queries to H_1, H_2, H_4, H_5, respectively.
2. *Phase 1:* Assume there are n users with identities $ID_1, \cdots, ID_n$ in the system. $\mathcal{B}$ randomly chooses $i^* \in [1, n]$ and performs the following steps.
   - *H_i-query (ID_i)* $(i = 1, 2, 4, 5)$*:* $\mathcal{B}$ performs as in the **Proof of Theorem 1**.

- *Master secret key query* (1^λ)*:* $\mathcal{B}$ returns α to $\mathcal{A}_3$.
- *Partial private key query* (ID_i)*:* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ then computes

$$D_i = (D_{i,1}, D_{i,2}) = (H_1^\alpha(ID), H_2^\alpha(ID)) = \left(g_2^{\alpha u_i}, g_2^{\alpha v_i}\right),$$

which can be computed with known g_2, α, u_i, v_i. $\mathcal{B}$ returns D_i to $\mathcal{A}_3$.
- *Secret key query* (ID_i)*:* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$, stores $(ID_i, x_i, -, -)$ into L_s, and returns x_i to $\mathcal{A}_3$.
- *Private key query* (ID_i)*:* For the i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i was not queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{\alpha x_i u_i}, g_2^{\alpha x_i v_i}\right),$$

which can be computed with known $g_2, \alpha, x_i, u_i, v_i$. It then stores $(ID_i, x_i, sk_i, -)$ into L_s and returns sk_i to $\mathcal{A}_3$.
- *Public key query* (ID_i)*:* For the i-th queried identity ID_i, if $ID_i \neq ID_{i^*}$, $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{\alpha x_i}, g_2^{x_i}, g_1^{x_i}\right),$$

It then stores $(ID_i, x_i, -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_3$;
If $ID_i \neq ID_{i^*}$, $\mathcal{B}$ randomly picks $x_i' \in \mathbb{Z}_p$, implicitly sets $x_i = ax_i'$, computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{\alpha a x_i'}, g_2^{a x_i'}, g_1^{a x_i'}\right),$$

which can be computed with known $g_1^a, g_2^a, \alpha, x_i'$. It then stores $(ID_i, x_i', -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_3$.
- *Proxy key query* (ID_{P_i})*:* For a proxy with the related item in L_s as (ID_{P_i}, x_{P_i}), $\mathcal{B}$ performs as a *public key query* step and obtains

$$pk_{P_i} = (X_{P_i}, Y_{P_i}, Z_{P_i}) = \left(g_1^{\alpha x_{P_i}}, g_2^{x_{P_i}}, g_1^{x_{P_i}}\right).$$

which can be computed with known $g_1, \alpha, g_2, x_{P_i}$. $\mathcal{B}$ then returns pk_{P_i} and sk_{P_i} to $\mathcal{A}_3$.

- *Token query (ID_i):* For a queried ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches L_s to find the related item of ID_i as (ID_i, x_i, sk_i), where $sk_i = (sk_1, sk_2)$. If the related item does not exist, $\mathcal{B}$ performs as follows. If ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{\alpha x_i u_i}, g_2^{\alpha x_i v_i}\right),$$

which can be computed with known g_2^a, x, u_i, v_i. It then stores (ID_i, x_i, sk_i) into L_s. $\mathcal{B}$ returns sk_2 to $\mathcal{A}_3$.

- *Proxy token query (ID_i, ID_{P_i}):* For a queried ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ performs as Aut step except that $\mathcal{B}$ computes $PI_i = H_2(ID_i)^{x_{P_i}} = g_2^{\alpha v_i x_i x_{P_i}}$, and the proxy token

$$\hat{tk}_i = H_2(ID)^{\alpha x + x \cdot x_P} = g_2^{\alpha v_i x_i + x_i x_{P_i}}$$

with known $g_2, \alpha, v_i, x_i, x_{P_i}$. $\mathcal{B}$ returns $\hat{tk}_i$ to $\mathcal{A}_3$.

3. *Challenge:* $\mathcal{A}_3$ sends $(s^*, ID^*, M_0^*, M_1^*)$ to $\mathcal{B}$, where s^* represents the designated challenge number, ID^* stands for the challenge identity, and two plaintexts M_0^* and M_1^* are selected from $\{0,1\}^\lambda$ with equal lengths. If $ID^* \neq ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, it randomly picks $w_1, w_2 \in \mathbb{Z}_p, R_1, R_2 \in \{0,1\}^{2l}, R_3 \in \{0,1\}^\lambda$ and implicitly sets $r_1 = w_1 c, r_2 = w_2 c$. It then computes the challenge ciphertext as

$$\begin{aligned}
C_1 &= g_1^{r_1} = g_1^{cw_1}, \\
C_2 &= R_1, \\
C_3 &= g_1^{r_2} = g_1^{cw_2}, \\
C_4 &= g_1^{x_{i^*} r_2} = g_1^{c x_{i^*} w_2}, \\
C_5 &= R_2, \\
C_6 &= R_3.
\end{aligned}$$

4. *Phase 2:* $\mathcal{B}$ interacts with $\mathcal{A}_3$ as *Phase 1* with the limitation that ID_{i^*} cannot be queried in *partial private key query*, *secret key query*, *private key query*, *token query*, and *proxy token query*.
5. Guess: $\mathcal{A}_3$ outputs its guess bit $\rho' \in \{0,1\}$.
6. *Solve:* $\mathcal{B}$ randomly chooses an item $(Q_i, H_4(Q_i) = \eta_i)$ from L_{H_4} and sets

$$e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*}' u_{i^*} w_1 \alpha}}$$

as the solution to the BDH instance.

7. *Analysis:* To successfully perform the reduction, the simulation should be indistinguishable from the real attack from the point of view of the adversary. As we can see, if the adversary chooses ID_{i^*} as the challenge identity, the simulation will not abort, which means that the simulation is indistinguishable from the real attack. The corresponding probability is $1/n$. Upon the case that the simulation is indistinguishable to the adversary, we have the following analysis. Since the adversary is assumed to break the security with advantage ϵ, we have that it issues the hash query $e(X_{i^*}, H_1(ID_{i^*}))^{r_1} = e(g_1^{\alpha a x'_{i^*}}, g_2^{bu_{i^*}})^{cw_1} = e(g_1, g_2)^{abc \cdot x'_{i^*} u_{i^*} w_1 \alpha}$ with probability ϵ. Thus, $\mathcal{B}$ finally can obtain the true solution to the given BDH instance as $e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x'_{i^*} u_{i^*} w_1 \alpha}}$ with probability $\frac{\epsilon}{q_{H_4}}$. In conclusion, $\mathcal{B}$ can successfully break the BDH assumption with probability

$$\Pr[\mathcal{B} \textit{ chooses the correct hash query} | \textit{the simulation is indistinguishable}] = \frac{\epsilon}{n \cdot q_{H_4}}.$$

□

Theorem 4. *For any PPT Type-IV adversary, our CLE-MET-PA scheme is OW-CPA secure based on the BDH assumption in the random oracle model.*

Proof of Theorem 4. Assume there exists an adversary $\mathcal{A}_4$ who can break the IND-CPA security of our scheme with a non-negligible advantage ϵ; we can construct a simulator $\mathcal{B}$ to break the BDH assumption. Given an instance as $(\mathcal{G}, g_1, g_1^a, g_1^c, g_2, g_2^a, g_2^b)$, $\mathcal{B}$ is to compute $e(g_1, g_2)^{abc}$ by running $\mathcal{A}_4$ as a subroutine. $\mathcal{B}$ and $\mathcal{A}_4$ play the following game.

1. *Setup:* $\mathcal{B}$ randomly picks a cryptographic hash function $H_3 : \{0,1\}^* \to \{0,1\}^\lambda$, randomly picks $\alpha \in \mathbb{Z}_p^*$, and sets the public parameter pp. $pp = \{\mathcal{G}, g_1, g_2, \bar{g}, H_1, H_2, H_3, H_4, H_5\}$, where $\bar{g} = g_1^\alpha$, H_1, H_2, H_4, H_5 are random oracles. pp is sent to $\mathcal{A}_4$. Lists $L_{H_1}, L_{H_2}, L_{H_4}, L_{H_5}, L_s$ are initially empty. Assume that $\mathcal{A}_4$ can make $q_{H_1}, q_{H_2}, q_{H_4}, q_{H_5}$ hash queries to H_1, H_2, H_4, H_5, respectively.
2. *Phase 1:* Assume there are n users with identities $ID_1, \cdots, ID_n$ in the system. $\mathcal{B}$ randomly chooses $i^* \in [1, n]$ and performs the following steps.
   - *H_i-query (ID_i)* ($i = 1, 2, 4, 5$)*:* $\mathcal{B}$ performs as in the **Proof of Theorem 2**.
   - *Master secret key query* (1^λ)*:* $\mathcal{B}$ returns α to $\mathcal{A}_4$.
   - *Partial private key query (ID_i):* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets
   $$H_1(ID_i) = g_2^{u_i},$$
   and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets
   $$H_2(ID_i) = g_2^{v_i},$$
   and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ then computes
   $$D_i = (D_{i,1}, D_{i,2}) = (H_1^\alpha(ID), H_2^\alpha(ID)) = \left(g_2^{\alpha u_i}, g_2^{\alpha v_i}\right),$$
   which can be computed with known g_2, α, u_i, v_i. $\mathcal{B}$ returns D_i to $\mathcal{A}_4$.
   - *Secret key query (ID_i):* For i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$, stores $(ID_i, x_i, -, -)$ into L_s, and returns x_i to $\mathcal{A}_4$.
   - *Private key query (ID_i):* For the i-th queried identity ID_i, if $ID_i = ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, if ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets
   $$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{\alpha x_i u_i}, g_2^{\alpha x_i v_i}\right),$$

which can be computed with known $g_2, \alpha, x_i, u_i, v_i$. It then stores $(ID_i, x_i, sk_i, -)$ into L_s and returns sk_i to $\mathcal{A}_4$.

- *Public key query (ID_i):* For i-th queried identity ID_i, if $ID_i \neq ID_{i^*}$, $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{\alpha x_i}, g_2^{x_i}, g_1^{x_i}\right),$$

It then stores $(ID_i, x_i, -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_4$;
If $ID_i \neq ID_{i^*}$, $\mathcal{B}$ randomly picks $x_i' \in \mathbb{Z}_p$, implicitly sets $x_i = ax_i'$, and computes

$$pk_i = (X_i, Y_i, Z_i) = \left(g_1^{\alpha a x_i'}, g_2^{a x_i'}, g_1^{a x_i'}\right),$$

which can be computed with known $g_1^a, g_2^a, \alpha, x_i'$. It then stores $(ID_i, x_i', -, pk_i)$ into L_s and returns pk_i to $\mathcal{A}_4$.

- *Proxy key query (ID_{P_i}):* For a proxy with the related item in L_s as (ID_{P_i}, x_{P_i}), $\mathcal{B}$ performs as *public key query* step and obtains

$$pk_{P_i} = (X_{P_i}, Y_{P_i}, Z_{P_i}) = \left(g_1^{\alpha x_{P_i}}, g_2^{x_{P_i}}, g_1^{x_{P_i}}\right).$$

which can be computed with known $g_1, \alpha, g_2, x_{P_i}$. $\mathcal{B}$ then returns pk_{P_i} and sk_{P_i} to $\mathcal{A}_4$.

- *Token query (ID_i):* For a queried ID_i, if $ID_i \neq ID_{i^*}$, $\mathcal{B}$ searches L_s to find the related item of ID_i as (ID_i, x_i, sk_i), where $sk_i = (sk_1, sk_2)$. If the related item does not exist, $\mathcal{B}$ performs as follows. If ID_i has not been queried to H_1, $\mathcal{B}$ randomly chooses u_i, sets

$$H_1(ID_i) = g_2^{u_i},$$

and stores $(u_i, ID_i, H_1(ID_i))$ into L_{H_1}. If ID_i has not been queried to H_2, $\mathcal{B}$ randomly chooses v_i, sets

$$H_2(ID_i) = g_2^{v_i},$$

and stores $(v_i, ID_i, H_2(ID_i))$ into L_{H_2}. $\mathcal{B}$ searches L_{H_1} and L_{H_2} to find the related items of ID_i as $(u_i, ID_i, H_1(ID_i))$ and $(v_i, ID_i, H_2(ID_i))$. $\mathcal{B}$ searches L_s and finds the related item of ID_i as $(ID_i, x_i, -, -)$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i \in \mathbb{Z}_p$ and computes

$$sk_i = (sk_{i,1}, sk_{i,2}) = \left(D_1^{x_i}, D_2^{x_i}\right) = \left(g_2^{\alpha x_i u_i}, g_2^{\alpha x_i v_i}\right),$$

which can be computed with known $g_2, \alpha, x_i, u_i, v_i$. It then stores (ID_i, x_i, sk_i) into L_s. $\mathcal{B}$ returns sk_2 to $\mathcal{A}_4$.
If $ID_i = ID_{i^*}$, $\mathcal{B}$ searches L_s to find the related item of ID_i as $(ID_i, x_i', -, -)$ and $(v_i, ID_i, H_2(ID_i))$. If the related item of ID_i does not exist, $\mathcal{B}$ randomly picks $x_i' \in \mathbb{Z}_p$ and computes

$$sk_2 = g_2^{\alpha a v_i x_i'},$$

which can be computed with the known g_2^a, α, v_i, x_i', $\mathcal{B}$ returns sk_2 to $\mathcal{A}_4$.

- *Proxy token query (ID_i, ID_{P_i}):* For a queried ID_i, $\mathcal{B}$ performs as Aut step except that $\mathcal{B}$ computes

$$PI_i = H_2(ID_i)^{x_{P_i}} = \begin{cases} g_2^{\alpha v_i x_i x_{P_i}}, & if\ i \neq i^*, \\ g_2^{\alpha a v_i x_i' x_{P_i}}, & if\ i = i^*. \end{cases}$$

$$\hat{tk} = H_2(ID)^{\alpha x + x \cdot x_P} = \begin{cases} g_2^{\alpha v_i x_i + x_i x_{P_i}}, & if\ i \neq i^*, \\ g_2^{\alpha a v_i x_i' + a x_i' x_{P_i}}, & if\ i = i^*. \end{cases}$$

with known $g_2, \alpha, v_i, x_i, x_i', x_{P_i}$.

3. *Challenge:* $\mathcal{A}_4$ sends (s^*, ID^*) to $\mathcal{B}$, where s^* represents the designated challenge number, and ID^* stands for the challenge identity. If $ID^* \neq ID_{i^*}$, $\mathcal{B}$ aborts. Otherwise, it chooses to randomly pick $M^* \in \{0,1\}^\lambda, w_1, w_2 \in \mathbb{Z}_p, R_1 \in \{0,1\}^{2l}$, and implicitly sets $r_1 = w_1 c, r_2 = w_2 c$. Taking as input M^*, s^*, it then iteratively computes $(f_0, f_1, \cdots, f_{s^*-1})$ and $f(x)$ same as $\mathsf{Enc}(pp, pk, pk_P, M, s)$ in Section 4.1. Then, it randomly picks $A \in \mathbb{Z}_p$, computes $f(A)$, before outputting the challenge ciphertext as follows:

$$\begin{aligned} C_1 &= g_1^{r_1} = g_1^{cw_1}, \\ C_2 &= R_1, \\ C_3 &= g_1^{r_2} = g_1^{cw_2}, \\ C_4 &= g_1^{x_{i^*}' r_2} = g_1^{c x_{i^*}' w_2}, \\ C_5 &= H_4\left(e(g_1^{cw_1}, g_2^{\alpha a v_{i^*} x_{i^*}'})\right) \oplus (A || f(A)), \\ C_6 &= H_5\left(s^* || C_1^* || C_2^* || C_3^* || C_4^* || C_5^* || e(g_1^{cw_1}, g_2^{\alpha a v_{i^*} x_{i^*}'}) || f_0 || f_1 || \cdots || f_{s^*-1}\right). \end{aligned}$$

4. *Phase 2:* $\mathcal{B}$ interacts with $\mathcal{A}_4$ as *Phase 1* with the limitation that ID_{i^*} cannot be queried in *partial private key query*, *secret key query*, and *private key query*.
5. Guess: $\mathcal{A}_4$ outputs its guess $M' \in \{0,1\}^\lambda$.
6. *Solve:* $\mathcal{B}$ randomly chooses an item $(Q_i, H_4(Q_i) = \eta_i)$ from L_{H_4} and sets

$$e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x_{i^*}' u_{i^*} w_1 \alpha}}$$

as the solution to the BDH instance.

7. *Analysis:* To successfully perform the reduction, the simulation should be indistinguishable from the real attack from the point of view of the adversary. As we can see, if the adversary chooses ID_{i^*} as the challenge identity, the simulation will not abort, which means the simulation is indistinguishable from the real attack. The corresponding probability is $1/n$. Upon the case that the simulation is indistinguishable to the adversary, we have the following analysis. Since the adversary is assumed to break the security with advantage ϵ, we have that it issues the hash query $e(X_{i^*}, H_1(ID_{i^*}))^{r_1} = e(g_1^{\alpha a x_{i^*}'}, g_2^{b u_{i^*}})^{cw_1} = e(g_1, g_2)^{abc \cdot x_{i^*}' u_{i^*} w_1 \alpha}$ with probability ϵ. Thus, $\mathcal{B}$ can finally obtain the true solution to the given BDH instance as

$e(g_1, g_2)^{abc} = Q_i^{\frac{1}{x'_{i^*} u_{i^*} w_1 \alpha}}$ with probability $\frac{\epsilon}{q_{H_4}}$. In conclusion, $\mathcal{B}$ can successfully break the BDH assumption with probability

$$\Pr[\mathcal{B} \textit{ chooses the correct hash query} | \textit{the simulation is indistinguishable}] = \frac{\epsilon}{n \cdot q_{H_4}}.$$

□

Theorem 5 (Number Security of CLE-MET-PA)**.** *In the information theoretical sense, no probabilistic polynomial-time adversary has a non-negligible advantage in breaking the number security of our CLE-MET-PA scheme.*

Proof of Theorem 5. In this game of number security, the adversary $\mathcal{A}_5$ tries to determine whether the underlying messages of t^* challenging ciphertexts $CT_1^*, \cdots, CT_{t^*}^*$ are equal or not with all the designated numbers of these ciphertexts, which are s^* and $s^* > t^*$. And, the underlying messages are chosen by $\mathcal{B}$ and they are unknown to $\mathcal{A}_5$. From the setting of our scheme, we have that $\mathcal{A}_5$ has two ways to check whether the underlying messages are equal or not, i.e., extracting M_i or $f_{i,j}$ of CT_i for some $j \in \{0, \cdots, t^* - 1\}$, for each $i \in \{1, \cdots, t^*\}$.

In line with the OW-CPA security proof, $\mathcal{A}_5$ has only a negligible advantage in obtaining the underlying message from the ciphertext under the BDH assumption. To extract a $f_{i,j}$ from the ciphertext $CT_i = (s, C_{i,1}, C_{i,2}, C_{i,3}, C_{i,4}, C_{i,5}, C_{i,6})$, $\mathcal{A}_5$ can make a *token query* or a *Proxy token query* on pk_i and obtain the corresponding token $tk_i = sk_{i,2}$ or $\hat{tk}_i = sk_{i,2} H_2(ID_i)^{x_i x_{P_i}}$ such that it is able to obtain the $A_i || f_i(A_i) = C_{i,5} \oplus H_4(e(C_{i,3}, tk_i))$ or $A_i || f_i(A_i) = C_{i,5} \oplus H_4\Big(e(C_{i,3}, \hat{tk}_i) / e(C_{i,4}, PI_i)\Big)$, where

$$f_i(A_i) = f_{i,0} + f_{i,1} A_i + \cdots + f_{i,t^*-1} A_i^{t^*-1}$$

and

$$f_{i,j} = H_3(M_i || s || f_{i,0} || \cdots || f_{i,j-1}),\ j \in \{0, \cdots, t^* - 1\}.$$

Obtaining $A_i || f_i(A_i)$ for each $i \in \{1, \cdots, t^*\}$, $\mathcal{A}_5$ has an equation set as

$$\begin{cases} f_1(A_1) = f_{1,0} + f_{1,1} A_1 + \cdots + f_{1,t^*-1} A_1^{t^*-1} \\ f_2(A_2) = f_{2,0} + f_{2,1} A_2 + \cdots + f_{2,t^*-1} A_2^{t^*-1} \\ \quad \vdots \\ f_{t^*}(A_{t^*}) = f_{t^*,0} + f_{t^*,1} A_{t^*} + \cdots + f_{t^*,t^*-1} A_{t^*}^{t^*-1} \end{cases},$$

where A_i for $i \in \{1, \cdots, t^* - 1\}$ are randomly chosen by $\mathcal{C}$. Therefore, after implicitly setting that $f_{i,k} = f_{j,k}$ for $i \in \{1, \cdots, t^* - 1\}, j \in \{0, \cdots, t^* - 1\}$, these $t^* - 1$ equations are nonlinearly correlated with each other by an overwhelming probability such that there are infinite solutions for $\{f_{i,j}\}_{1 \le i \le t^*, 0 \le j \le t^*-1}$. In addition, $\mathcal{A}_5$ has no other information to further ensure whether the underlying messages are equal or not. It can only randomly guess b'. In conclusion, the adversary has a negligible advantage in breaking the number security of the proposed CLE-MET-PA scheme. □

6. Performance Analysis and Extension

6.1. Performance Analysis of CLE-MET-PA

We conducted a visual comparison between our scheme and several existing schemes [15,30,31,40], and the results are presented in Table 2. Among them, Refs. [15,30,40] only support pairwise equality tests. Therefore, in scenarios where we need to test the equality of s ciphertexts, these three schemes require $s - 1$ executions of their test algorithms.

We adopted the commonly used approach for the performance analysis of most PKEET schemes, which involves calculating the complexity of significant algorithms, ciphertext

size, and scheme-specific functionalities. This method allows us to evaluate different algorithms using the same standard across various platforms. Initially, we evaluated the complexity of encryption, decryption, and test algorithms, primarily focusing on three metrics: the number of exponential operations, hash operations, and bilinear pairings, denoted by E, H, and P, respectively. We excluded efficient operations such as addition, multiplication, and XOR. Next, we considered additional metrics such as ciphertext size and three functionalities: the anti-key management feature, the anti-key escrow feature, and support for proxy-assisted authorization.

As shown in Table 2, our scheme demonstrates significant advantages in terms of the multi-ciphertext equality test algorithm. In comparison to traditional schemes that support pairwise equality tests, our multi-ciphertext equality test exhibits lower computational complexity than conducting multiple pairwise equality tests. Specifically, even in extreme situations where all tokens are proxy tokens, we only require 2s of the most computationally expensive bilinear pairings. This is notably lower than other schemes that support pairwise equality tests. Furthermore, our scheme inherently possesses certificateless encryption properties, effectively addressing the key management problem in PKEET and the key escrow problem in IBEET. This makes our scheme more suitable in scenarios where certificate management is challenging or where anonymous communication is required. Additionally, we introduced the feature of proxy-assisted authorization, enabling users to delegate proxy authorization while they are offline and better protect their private keys. These characteristics make our CLE-MET-PA scheme more competitive for practical use.

Table 2. Comparison among several equality test schemes.

Schemes	**[15]**	**[30]**	**[40]**	**[31]**	**Ours**
Enc	6E+3H+2P	5E+4H+4P	2E+3H+3P	$(s+2)$E+$(s+3)$H	$(s+3)$E+$(s+5)$H+6P
Dec	4E+3H+2P	2E+4H+2P	1E+2H+3P	$(s+1)$E+$(s+2)$H	$(s-1)$E+$(s+3)$H+2P
Test	$(s-1)$(2E+4P)	$(s-1)$(2H+4P)	$(s-1)$(4H+4P)	sE+ 2sH+SE	2sH+$(s+j)$P+SE
$\|CT\|$	$5\|\mathbb{G}\|+\|\mathbb{Z}_p\|$	$3\|\mathbb{G}\|+\|\mathbb{Z}_p\|+\{0,1\}^\lambda$	$3\|\mathbb{G}\|+\|\mathbb{Z}_p\|$	$2\|\mathbb{G}\|+5\|\mathbb{Z}_p\|+\{0,1\}^\lambda$	$3\|\mathbb{G}\|+5\|\mathbb{Z}_p\|+\{0,1\}^\lambda$
AntiKM	✓	✓	✓	×	✓
AntiKE	×	✓	✓	✓	✓
PA	×	×	×	×	✓

E, H, P, and SE represent the computation cost of an exponential operation, hash operation, pairing operation, and solving an equation set, respectively. $|\mathbb{Z}_p|, |\mathbb{G}|$ represent the bit length of a group element in $\mathbb{Z}_p, \mathbb{G}$, respectively. j: number of proxy tokens, $(0 \leq j \leq s)$. s: number of ciphertexts to be verified. AntiKM, AntiKE, and PA represent anti-key management feature, anti-key escrow feature and proxy-assisted authorization feature, respectively. The presence of a checkmark indicates that the feature has been included, while a cross indicates that the feature has not been included.

6.2. Extension

As mentioned before in this paper, our CLE-MET-PA scheme achieves IND-CPA security against adversaries without a trapdoor and OW-CPA security against adversaries with a trapdoor. We extend our scheme to CCA security with FO transformation [32,33] by simple modifications. The improved version of CLE-MET-PA is as follows.

Setup(1^λ): Almost the same as Setup(1^λ) in Section 4.1. The difference is that while generating pp and msk, additionally generate a cryptographic hash function $H_6 : \{0,1\}^* \rightarrow \{0,1\}^l$, and add it in pp.

Partial-Private-Key-Extract (pp, msk, ID): Same as Partial-Private-Key-Extract(pp, msk, ID) in Section 4.1.

Set-Secret-Value (pp, ID): Same as Set-Secret-Value (pp, ID) in Section 4.1.

Set-Private-Key (pp, D, x): Same as Set-Private-Key (pp, D, x) in Section 4.1.

Set-Public-Key (pp, x): Same as Set-Public-Key (pp, x) in Section 4.1.

Set-Proxy-Key (pp, ID_P): Same as Set-Proxy-Key (pp, ID_P) in Section 4.1.

$\mathsf{Enc}(pp, pk, M, s)$: Taking as input the system parameter pp, a user public key pk, and a proxy public key pk_P, check whether $e(X, g_2) = e(\bar{g}, Y)$ holds; if not, output $\perp$ and abort. Then, taking as input a message $M \in \mathbb{Z}_p$, and a number $s \in \mathbb{Z}_p$, iteratively compute $(f_0, f_1, \cdots, f_{s-1})$ and $f(x)$ same as $\mathsf{Enc}(pp, pk, M, s)$ in Section 4.1.

It randomly chooses $A, r_1, r_2 \in \mathbb{Z}_p$, computes $C_3 = H_4(r_1) \oplus (M||r_1)$, $R = H_3(r_1||M||C_3)$, and outputs the ciphertext $CT = (s, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ as

$$\begin{aligned}
&C_1 = g_1^R, \quad C_2 = H_6\Big(e(X, H_1(ID))^R\Big) \oplus r_1,\\
&C_3 = H_4(r_1) \oplus (M||r_1),\\
&C_4 = g_1^{r_2}, \quad C_5 = Z^{r_2},\\
&C_6 = H_4(e(X, H_2(ID))^{r_2}) \oplus (A||f(A)),\\
&C_7 = H_5(s||C_1||C_2||C_3||C_4||C_5||C_6||e(X, H_2(ID))^{r_2}||f_0||f_1||\cdots||f_{s-1}).
\end{aligned}$$

$\mathsf{Dec}(CT, sk)$: Taking as input a ciphertext $CT = (s, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ and a secret key $sk = (H_1(ID)^{\alpha x}, H_2(ID)^{\alpha x})$, the decrypt algorithm computes

$$r_1' = C_2 \oplus H_6(e(sk_1, C_1)), \quad M'||r_1' = C_3 \oplus H_4(r_1'), \quad R' = H_3(r_1'||M'||C_3).$$

It then computes

$$\begin{aligned}
&f_0' = H_3(M'||s), \quad f_1' = H_3(M'||s||f_0'), \quad \cdots,\\
&f_{s-1}' = H_3(M'||s||f_0'||\cdots||f_{s-2}')
\end{aligned}$$

and checks whether the following equations hold or not

$$\begin{aligned}
&C_1 = g_1^{R'},\\
&f'(A') = f_0' + f_1'A' + \cdots + f_{s-1}'A'^{s-1},\\
&C_7 = H_5(s||C_1||C_2||C_3||C_4||C_5||C_6||e(sk_2, C_4)||f_0'||f_1'||\cdots||f_{s-1}'),
\end{aligned}$$

where $A'||f'(A') = C_5 \oplus H_4(e(sk_2, C_4))$. If all the equations hold, it returns

$$M = M'.$$

Otherwise, it returns $\perp$.

The structures of the subsequent Aut, Proxy-Aut, and Test algorithms are essentially the same as the relevant algorithms in Section 4.1. Since these algorithms do not affect the security of the scheme, we will not elaborate further.

It is important to highlight that the enhanced scheme offers improved security, albeit with a slight increase in system parameter size, cipher size, and computation for encryption and decryption algorithms. In particular, the system parameters (pp) now incorporate an additional hash function. Furthermore, the cipher size increased by $2|\mathbb{Z}_p|$, and both the encryption and decryption algorithms entail two additional hash operations (denoted by 2H). By definition of FO transformation, our improved CLE-MET-PA scheme achieves IND-CCA security against adversaries without the trapdoor of the challenge ciphertext and OW-CCA security against adversaries with trapdoor of the challenge ciphertext.

7. Conclusions

In this work, we presented the notion of CLE-MET-PA, which integrates the characteristics of solving the key management problem in the public key encryption cryptosystem and addressing the key escrow problem in the IBE cryptosystem. It combines the feature of the multi-ciphertext equality test and supports proxy-assisted authorization, enhancing the practical use of the scheme. We formalized the system model and five security models for CLE-MET-PA, and proved that our scheme achieves OW-CPA/IND-CPA security. Finally,

through an FO transformation, we obtained the improved version of CLE-MET-PA, which achieves OW-CCA/IND-CCA security.

Author Contributions: Conceptualization, S.D. and Z.Z.; methodology, S.D. and Z.Z.; writing—original draft preparation, S.D.; writing—review and editing, W.G. and S.Z.; supervision, B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 61972457, 62102299, 62002288, U19B2021, 62272362, 62202363, and the Youth Innovation Team of Shaanxi Universities, Science and Technology on Communication Security Laboratory Foundation (61421030202012103).

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qian, L.; Luo, Z.; Du, Y.; Guo, L. Cloud computing: An overview. In Proceedings of the Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, 1–4 December 2009; Proceedings 1; Springer: Berlin/Heidelberg, Germany, 2009; pp. 626–631.
2. Dillon, T.; Wu, C.; Chang, E. Cloud computing: Issues and challenges. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, Australia, 20–23 April 2010; pp. 27–33.
3. Zhang, C.; Hu, C.; Wu, T.; Zhu, L.; Liu, X. Achieving Efficient and Privacy-Preserving Neural Network Training and Prediction in Cloud Environments. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 4245–4257. [CrossRef]
4. Hu, C.; Zhang, C.; Lei, D.; Wu, T.; Liu, X.; Zhu, L. Achieving Privacy-Preserving and Verifiable Support Vector Machine Training in the Cloud. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3476–3491. [CrossRef]
5. Goldreich, O.; Ostrovsky, R. Software protection and simulation on oblivious RAMs. *J. ACM (JACM)* **1996**, *43*, 431–473. [CrossRef]
6. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, S&P 2000, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
7. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
8. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In Proceedings of the Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Proceedings 23; Springer: Berlin/Heidelberg, Germany, 2004; pp. 506–522.
9. Yang, G.; Tan, C.H.; Huang, Q.; Wong, D.S. Probabilistic public key encryption with equality test. In Proceedings of the Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, 1–5 March 2010; Proceedings; Springer: Berlin/Heidelberg, Germany, 2010; pp. 119–131.
10. Tang, Q. Public key encryption supporting plaintext equality test and user-specified authorization. *Secur. Commun. Netw.* **2012**, *5*, 1351–1362. [CrossRef]
11. Tang, Q. Towards public key encryption scheme supporting equality test with fine-grained authorization. In Proceedings of the Australasian Conference on Information Security and Privacy, ACISP 2011, Melbourne, Australia, 11–13 July 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 389–406.
12. Huang, K.; Tso, R.; Chen, Y.C.; Rahman, S.M.M.; Almogren, A.; Alamri, A. PKE-AET: Public key encryption with authorized equality test. *Comput. J.* **2015**, *58*, 2686–2697. [CrossRef]
13. Ma, S.; Zhang, M.; Huang, Q.; Yang, B. Public key encryption with delegated equality test in a multi-user setting. *Comput. J.* **2015**, *58*, 986–1002. [CrossRef]
14. Ma, S.; Huang, Q.; Zhang, M.; Yang, B. Efficient public key encryption with equality test supporting flexible authorization. *IEEE Trans. Inf. Forensics Secur.* **2014**, *10*, 458–470. [CrossRef]
15. Ma, S. Identity-based encryption with outsourced equality test in cloud computing. *Inf. Sci.* **2016**, *328*, 389–402. [CrossRef]
16. Lee, H.T.; Ling, S.; Seo, J.H.; Wang, H. Semi-generic construction of public key encryption and identity-based encryption with equality test. *Inf. Sci.* **2016**, *373*, 419–440. [CrossRef]
17. Wu, T.; Ma, S.; Mu, Y.; Zeng, S. ID-based encryption with equality test against insider attack. In Proceedings of the Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, 3–5 July 2017; Proceedings, Part I 22; Springer: Berlin/Heidelberg, Germany, 2017; pp. 168–183.
18. Wu, L.; Zhang, Y.; Choo, K.K.R.; He, D. Efficient and secure identity-based encryption scheme with equality test in cloud computing. *Future Gener. Comput. Syst.* **2017**, *73*, 22–31. [CrossRef]
19. Alornyo, S.; Asante, M.; Hu, X.; Mireku, K.K. Encrypted traffic analytic using identity based encryption with equality test for cloud computing. In Proceedings of the 2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST), Accra, Ghana, 22–24 August 2018; pp. 1–4.

20. 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. [CrossRef]
21. Liao, Y.; Fan, Y.; Liang, Y.; Liu, Y.; Mohammed, R. Cryptanalysis of an identity-based encryption scheme with equality test and improvement. *IEEE Access* **2019**, *7*, 75067–75072. [CrossRef]
22. Ling, Y.; Ma, S.; Huang, Q.; Xiang, R.; Li, X. Group id-based encryption with equality test. In Proceedings of the Information Security and Privacy: 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, 3–5 July 2019; Proceedings 24; Springer: Berlin/Heidelberg, Germany, 2019; pp. 39–57.
23. Ming, Y.; Wang, E. Identity-based encryption with filtered equality test for smart city applications. *Sensors* **2019**, *19*, 3046. [CrossRef]
24. Susilo, W.; Duong, D.H.; Le, H.Q. Efficient post-quantum identity-based encryption with equality test. In Proceedings of the 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), Hong Kong, 2–4 December 2020; pp. 633–640.
25. Alornyo, S.; Zhao, Y.; Zhu, G.; Xiong, H. Identity Based Key-Insulated Encryption with Outsourced Equality Test. *Int. J. Netw. Secur.* **2020**, *22*, 257–264.
26. Ramadan, M.; Liao, Y.; Li, F.; Zhou, S.; Abdalla, H. IBEET-RSA: Identity-based encryption with equality test over RSA for wireless body area networks. *Mob. Networks Appl.* **2020**, *25*, 223–233. [CrossRef]
27. Lin, X.J.; Wang, Q.; Sun, L.; Qu, H. Identity-based encryption with equality test and datestamp-based authorization mechanism. *Theor. Comput. Sci.* **2021**, *861*, 117–132. [CrossRef]
28. Zhu, H.; Xue, Q.; Li, T.; Xie, D. Traceable Scheme of Public Key Encryption with Equality Test. *Entropy* **2022**, *24*, 309. [CrossRef]
29. Al-Riyami, S.S.; Paterson, K.G. Certificateless public key cryptography. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 30 November–4 December 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 452–473.
30. Qu, H.; Yan, Z.; Lin, X.J.; Zhang, Q.; Sun, L. Certificateless public key encryption with equality test. *Inf. Sci.* **2018**, *462*, 76–92. [CrossRef]
31. Susilo, W.; Guo, F.; Zhao, Z.; Wu, G. PKE-MET: Public-key encryption with multi-ciphertext equality test in cloud computing. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1476–1488. [CrossRef]
32. Fujisaki, E.; Okamoto, T. Secure integration of asymmetric and symmetric encryption schemes. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 537–554. [CrossRef]
33. Fujisaki, E.; Okamoto, T. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptol.* **2013**, *26*, 80–101. [CrossRef]
34. Boneh, D.; Boyen, X.; Shacham, H. Short group signatures. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 41–55.
35. Smart, N.P.; Vercauteren, F. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discret. Appl. Math.* **2007**, *155*, 538–547. [CrossRef]
36. Galbraith, S.D.; Paterson, K.G.; Smart, N.P. Pairings for cryptographers. *Discret. Appl. Math.* **2008**, *156*, 3113–3121. [CrossRef]
37. Chatterjee, S.; Menezes, A. On cryptographic protocols employing asymmetric pairings—The role of Ψ revisited. *Discret. Appl. Math.* **2011**, *159*, 1311–1322. [CrossRef]
38. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 213–229.
39. Boyen, X.; Mei, Q.; Waters, B. Direct chosen ciphertext security from identity-based techniques. In Proceedings of the 12th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 7–11 November 2005; pp. 320–329.
40. Zhao, M.; Ding, Y.; Tang, S.; Liang, H.; Yang, C.; Wang, H. Dual-server certificateless public key encryption with authorized equality test for outsourced IoT data. *J. Inf. Secur. Appl.* **2023**, *73*, 103441. [CrossRef]