Public Key Encryption with Delegated Equality Test in a Multi-User Setting

SHA MA¹, MINGWU ZHANG²,³, QIONG HUANG¹,* AND BO YANG⁴

¹College of Informatics, South China Agricultural University, Guangzhou, Guangdong, China
²School of Computers, Hubei University of Technology, Wuhan, Hubei, China
³State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
⁴School of Computer Science, Shaanxi Normal University, Shaanxi, Xi’an, China
*Corresponding author: csqhuang@gmail.com

Probabilistic public key encryption with equality test (PKEET), introduced by Yang et al. in CT-RSA 2010, is able to check whether two ciphertexts are encryptions of the same message under different public keys without leaking anything else about the message encrypted under either public key. PKEET schemes have many applications, for example, in constructing searchable encryption and partitioning encrypted data. Previous PKEET schemes lack a delegation mechanism for users to specify who can perform the equality test between their ciphertexts. In this paper, we propose the notion of public key encryption with delegated equality test (PKE-DET), which requires only the delegated party to deal with the work in a practical multi-user setting, and present a concrete construction in Type 2 pairing, which is provably secure under the newly introduced security notions.

Keywords: searchable encryption; delegated equality test; multi-user setting; type 2 pairing

Received 6 November 2013; revised 26 February 2014
Handling editor: Chris Mitchell

1. INTRODUCTION

The Database-as-a-Service (DaaS) model [1] is a new computing paradigm in cloud computing. Since highly sensitive data are now stored in locations without the data owner’s control, such as leased space and partners’ sites, it puts data confidentiality at risk. Therefore, such a varying trust scenario necessitates encryption techniques in the context of outsourced database [2–4]. In the database application equal join as a basic operator is used to concatenate each tuple of the first table with each tuple of the second table and output only those pairs of tuples with the same value on those joined attributes. We observe that such an operator in the DaaS model necessitates a public key encryption (PKE) to support equality test on ciphertexts without decryption. Those ciphertexts may be generated for a single user [5, 6], or more generally, for multiple users [7]. Motivated by this intuition, we propose a PKE with delegated equality test (PKE-DET) under multiple public keys, which can be described as follows: given any two ciphertexts $c_A$ and $c_B$ generated under Alice’s public key $pk_A$ and Bob’s public key $pk_B$, respectively, the third party with key pair $(pk_S, sk_S)$ who is delegated to perform the equality test on their ciphertexts can use two trapdoors $t_A$ and $t_B$ from Alice and Bob to evaluate the function $\text{Test}(c_A, t_A, c_B, t_B, sk_S)$, which returns 1 if and only if $c_A$ and $c_B$ are encryptions of the same message regardless of $pk_A = pk_B$ or not.

PKE-DET and PRE. Blaze et al. [8] formulated the concept of proxy re-encryption (PRE) cryptosystem and proposed the first bidirectional PRE scheme. Subsequently, different PRE schemes with various properties [9–13] were proposed. In a PRE scheme, a proxy is given special information that allows it to translate a ciphertext under one key into a ciphertext of the same message under a different key. The proxy cannot, however, learn anything about the message encrypted under either key. PRE schemes have many practical applications, including distributed storage, email and digital rights management (DRM) [10].

Both PKE-DET and PRE are concerned with ciphertexts of the same plaintexts under different public keys but their functionalities are different. Without decryption PKE-DET is used to check whether two ciphertexts are the encryptions of the same plaintext. PRE can be used to transform a ciphertext to another ciphertext with the same message.

PKE-DET and PEKS. PKE with keywords search (PEKS) was first proposed by Boneh et al. [14] and studies searching
keywords on encrypted data. A sender makes a ciphertext of a keyword \( w \) by using the receiver’s public key and sends it to the server. A receiver makes a trapdoor \( t_w \) for a keyword \( w \) and uploads it to the server, which tests if \( w = w' \). Several PEKS schemes with additional functionalities \([15–22]\) have been proposed thus far.

Both PEKS and PKE-DET can be regarded as a category of PKE supporting search on ciphertexts. But the differences between PEKS and PKE-DET include the following:

1. PEKS ciphertexts are generated under the same public key and the only trapdoor is generated under the corresponding private key. However, PKE-DET ciphertexts are generated under different public keys and the unencrypted trapdoor for each receiver is generated under its private key.

2. PKE-DET supports keywords search trivially provided by PEKS. The trapdoor consists of both a ciphertext \( c \) of keyword \( w \) and the associated trapdoor \( t_g \) in PKE-DET. Then, given a ciphertext \( c' \) of keyword \( w' \), the server with secret key \( sk_S \) tests whether \( \text{Test}(c, t_g, c', t_g, sk_S) \) is equal to 1.

3. The server has the controllability with different granularity over the delegated search capability for PEKS and PKE-DET. PEKS is used to check if the ciphertext contains a particular keyword; it can be viewed as an implementation of fine-grained control. However, PKE-DET is used to search the ciphertext of a user for any keyword as long as it is given its trapdoor. It is viewed as an implementation of coarse-grained control.

**PKE-DET and PKEET.** PKE with equality test (PKEET) \([7]\) was firstly proposed to check whether two ciphertexts encrypted under different public keys contain the same message. However, their formulation lacks an authorization mechanism to specify who can perform equality test between their ciphertexts. In fact, any entity can perform the test in PKEET. Later, to mitigate the potential vulnerabilities, Tang \([23]\) integrated a fine-grained authorization policy enforcement mechanism into PKEET and proposed an enhanced primitive, namely FG-PKEET. Also, Tang \([24]\) proposed an all-or-nothing PKE (AoN-PKEET), which introduces a coarse-grained authorization capability to specify who can perform a plaintext equality test from their ciphertexts, Tang \([25]\) extended FG-PKEET to a two-proxy setting, where two proxies need to collaborate in order to perform the equality test.

PKE-DET can also be viewed as an extension of PKEET. It is similar to the AoN-PKEET except for security enhancement. Our PKEET variant has a modest authorization mechanism in many database applications. For example, in a hospital scenario (Fig. 1) suppose that a hospital has many branches distributed in different regions. Each branch has its public/private key pair released by the hospital. The table for each branch (e.g., \( \text{PatA} \) and \( \text{PatB} \) tables for branch A and B, respectively) contains the patient ID, age and encrypted disease name, denoted by \( \text{PatId} \), \( \text{Age} \) and \( \text{Enc(DisName)} \), respectively. To find the average age of patients in the department of pediatrics in the \( \text{PatA} \) table with the same disease in the department of pediatrics in the \( \text{PatB} \) table, the SQL clause can be written as follows:

\[
\text{SELECT AVG(PatA.Age) FROM } \text{PatA}, \text{PatB WHERE } \text{PatA.Enc(DisName)} = \text{PatB.Enc(DisName)} \text{ AND } \text{PatA.department} = '\text{pediatrics}' \text{ AND } \text{PatB.department} = '\text{pediatrics}'
\]

Note that a special operator \( = \) appears in this clause to denote the equality test on ciphertexts without the decryption. Owing to this new operator, a query executor may need to be reconstructed. If all values in the \( \text{PatA.Enc(DisName)} \) column and the \( \text{PatB.Enc(DisName)} \) column are encrypted by PKE-DET under different public keys \( \text{pkA} \) and \( \text{pkB} \), respectively, the database server can execute the operation after receiving the trapdoors from the branch A and the branch B.

**Our contribution.** We propose the notion of PKE-DET, which owns the following properties:

1. Unlike PEKS in which only the metadata part of the ciphertext is searchable, the encrypted data are both searchable and decryptable in our scheme.

2. If the server is not delegated the equality test on ciphertexts, it cannot deduce any meaningful information from the encrypted data.

3. Once the data owner delegates the equality test on ciphertexts to the server, it can go offline. Namely, after the server obtains the trapdoors from the data owner, without decryption it can test by itself whether their ciphertexts contain the same message.

We propose a scheme implemented in the Type 2 pairing \([26]\) and prove its security against three types of adversaries in the random oracle model. This scheme is shown to satisfy:

1. **One-Wayness under a Chosen Ciphertext Attack (OW-CCA) Against Type-I adversary.** The delegated server with the trapdoor can perform ciphertext comparison, so the scheme cannot satisfy the standard level of ciphertext security and only can achieve OW-CCA security.

2. **Indistinguishable encryption under a Chosen Ciphertext Attack (IND-CCA) Against Type-II or Type-III adversary.** For the delegated server without getting the trapdoor or for the outsider who can steal the trapdoor, the scheme can achieve the standard level of ciphertext security: IND-CCA security.

3. **Anonymous Trapdoor (Anon-TD).** Since each receiver’s unencrypted trapdoor is only related to its private key, the scheme cannot reveal the corresponding receiver given the trapdoor. We call it Anon-TD security.
FIGURE 1. A scenario for the PKE-DET scheme.

Organization. The rest of this paper is organized as follows. In Section 2, we give some preliminaries. In Section 3, we formally define PKE-DET and give its security definitions. Next, we provide a construction of PKE-DET in Section 4 and its security proofs in Section 5. In Section 6, we compare it with other schemes. Section 7 concludes.

2. PRELIMINARIES

Bilinear map: Let $G_1$, $G_2$ and $G_T$ be three multiplicative cyclic groups of prime order $p$. Suppose that $g_1$ and $g_2$ are generators of $G_1$ and $G_2$, respectively. A bilinear map $e : G_1 \times G_2 \rightarrow G_T$ satisfies the following properties:

1. Bilinear: For any $g_1 \in G_1$, $g_2 \in G_2$ 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: There is an efficient algorithm to compute $e(g_1, g_2)$ for any $g_1 \in G_1$ and $g_2 \in G_2$.

If $G_1 \neq G_2$, then the pairing is asymmetric. In the asymmetric setting, if there is an efficiently computable isomorphism $\psi : G_2 \rightarrow G_1$, then $e$ is called a Type 2 pairing [26, 27].

Using asymmetric pairing is crucial for the security of our scheme. With symmetric pairing ($G_1 = G_2$), the decisional Diffie–Hellman (DDH) problem is not intractable in $G_1$ (or $G_2$). However, we use the DDH assumption in $G_1$ in the security proof.

External Diffie–Hellman (XDH) assumption on $(G_1, G_2)$: The XDH assumption implies the existence of two groups $G_1, G_2$ ($G_1 \neq G_2$) with the following properties:

1. the discrete logarithm problem, the computational Diffie–Hellman problem (CDH) and the computational co-Diffie–Hellman problem (co-DH) are all intractable in $G_1$ and $G_2$;
2. there exists an efficiently computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$;
3. the DDH problem is intractable in $G_1$;

where the CDH problem, the DDH problem and the co-DH problem are defined as follows.

CDH problem on $G_1$: We say that the CDH problem is $\epsilon$-hard in $G_1$, if given 3-tuple $(g_1, g_1^a, g_1^b) \in G_1^3$ as input and any randomized algorithm $A$ computes $g_1^{ab}$ with advantage:

$$\text{Adv}_{\text{CDH},G_1} = \Pr[A(g_1, g_1^a, g_1^b) = g_1^{ab}] \leq \epsilon.$$

We say that the CDH assumption holds if for any polynomial-time algorithm $A$, its advantage $\text{Adv}_{\text{CDH},G_1}$ is negligible.

DDH problem on $G_1$: We say that the DDH problem is $\epsilon$-hard in $G_1$, if given two 4-tuples $X = (g_1, g_1^a, g_1^b, g_1^{ab}) \in G_1^4$ and $Y = (g_1, g_1^c, g_1^d, g_1^{cd}) \in G_1^4$ as input, and any randomized algorithm $A$ distinguishes the random variables $X$ and $Y$ with advantage:

$$\text{Adv}_{\text{DDH},G_1} = |\Pr[A(X) = 1] - \Pr[A(Y) = 1]| \leq \epsilon.$$
We say that the DDH assumption holds if, for any polynomial-time algorithm \( A \), its advantage \( \text{Adv}^{\text{DDH}}_{A,G_1,G_2,G_3} \) is negligible.

**co-DH problem on \((G_1, G_2)\):** We say that the co-DH problem is \( \epsilon \)-hard in \( G_1 \) and \( G_2 \) if given \((g_1, g_1^\epsilon) \in G_1^2 \) and \((g_2, g_2^\epsilon) \in G_2^2 \) as input, where \( g_1 = \psi(g_2) \), and any randomized algorithm \( A \) computes \( g_2^\epsilon \) with advantage:

\[
\text{Adv}^{\text{co-DH}}_{A,G_1,G_2} = \text{Pr}[A(g_1, g_2, g_2^\epsilon) = g_2^\epsilon] \leq \epsilon.
\]

We say that the co-DH assumption holds if, for any polynomial-time algorithm \( A \), its advantage \( \text{Adv}^{\text{co-DH}}_{A,G_1,G_2,G_3} \) is negligible.

### 3. DEFINITIONS

In the PKE-DET scheme, there are three roles: the data owner, the delegated party, and the cloud server. The data owner wants to store some sensitive data in the cloud server. Initially, it encrypts these sensitive data using its public key to protect the data privacy. Some day, if it wants to delegate equality test to store some sensitive data in the cloud server, it can use its private key and the server’s public key to generate a trapdoor. With this trapdoor, it can keep this trapdoor or give it to the delegated party, who also needs to retrieve encrypted data on the cloud server. After the server receives a query from the data owner or the delegated party, it then uses the trapdoors to search ciphertexts. Finally, the delegated party obtains the returned matched ciphertexts. Figure 2 illustrates the overview of the PKE-DET scheme. Formally, we give the following definitions.

**Definition 3.1.** PKE-DET scheme consists of the following algorithms:

1. **\( \text{Setup}(\lambda) \):** This setup algorithm takes as input a security parameter \( \lambda \) and outputs public parameters \( pp \).
2. **\( \text{KeyGen}_A(pp) \):** This key generation algorithm takes as input the public parameters \( pp \) and outputs the server’s public/private key pair \((pk_A, sk_A)\).
3. **\( \text{KeyGen}_B(pp) \):** This key generation algorithm takes as input the public parameters \( pp \) and outputs the public/private key pair \((pk_B, sk_B)\) for the data owner A.
4. **\( \text{Encrypt}(pk_A, m) \):** This encryption algorithm takes as input A’s public key \( pk_A \) and a message \( m \in \text{M} \) where \( \text{M} \) is the plaintext space, and outputs A’s ciphertext \( c_A \).
5. **\( \text{Decrypt}(sk_A, c_A) \):** This decryption algorithm is run by the data owner, which takes as input A’s secret key \( sk_A \) and ciphertext \( c_A \), and outputs a plaintext \( m \) or \( \bot \) indicating failure.
6. **\( \text{Delegate}(sk_A, pk_B) \):** This delegate algorithm is run by the data owner, which takes as input A’s private key \( sk_A \) and the server’s public key \( pk_B \), and outputs the trapdoor \( t_A \) for A to test on its ciphertexts by the server.
7. **\( \text{Test}(c_A, t_A, c_B, t_B, sk_S) \):** This test algorithm is run by the server, which takes as input A’s ciphertext \( c_A \), a trapdoor \( t_A \) associated with A, another data owner B’s ciphertext \( c_B \), a trapdoor \( t_B \) associated with B, and the server’s private key \( sk_S \), and outputs \( 1 \) if the two ciphertexts are encryptions of the same message, otherwise outputs \( 0 \).

**Soundness.** The soundness property of the PKE-DET scheme consists of the following three conditions: (1) The encryption/decryption functionality works well, which means ciphertexts generated by the Encrypt algorithm can be decrypted correctly by the Decrypt algorithm. (2) If two ciphertexts are decrypted to the same message by the Decrypt algorithm, the Test algorithm certainly returns 1. (3) If two ciphertexts are decrypted to different messages by the Decrypt algorithm, the probability of the event that the Test algorithm returns 1 is negligible.

**Definition 3.2.** A PKE-DET scheme is sound if, for any \( pp \leftarrow \text{Setup}(\lambda) \), \((pk_S, sk_S) \leftarrow \text{KeyGen}_S(pp) \), \((pk_A, sk_A) \leftarrow \text{KeyGen}_A(pp) \), and \((pk_B, sk_B) \leftarrow \text{KeyGen}_B(pp) \), the following conditions are satisfied.

1. For any \( m \in \text{M} \), \( \text{Decrypt}(\text{Encrypt}(m, pk_A), sk_A) = m \) always holds.
2. For any ciphertexts \( c_A \) and \( c_B \), if \( \text{Decrypt}(c_A, sk_A) = \text{Decrypt}(c_B, sk_B) \) and \( \text{Decrypt}(c_A, sk_A) \neq \bot \), \( \text{Test}(c_A, t_A, c_B, t_B, sk_S) = 1 \) always holds.
3. For any ciphertexts \( c_A \) and \( c_B \), if \( \text{Decrypt}(c_A, sk_A) \neq \bot \), \( \text{Decrypt}(c_B, sk_B) \neq \bot \) and \( \text{Decrypt}(c_A, sk_A) \neq \text{Decrypt}(c_B, sk_B) \), \( \text{Pr}[\text{Test}(c_A, t_A, c_B, t_B, sk_S) = 1] \leq \epsilon(\lambda) \) where a function \( \epsilon \) is negligible in input parameter \( \lambda \).

As to the ciphertext security of the PKE-DET scheme, we consider four types of adversaries whose main goal is to reveal information about the encrypted data.

1. Type-I adversary represents a curious server, who owns both the server’s private key and the trapdoor. With respect to such an adversary, we define the notion of OW-CCA security.
2. Type-II adversary represents a curious server, who owns the server’s private key but does not obtain the trapdoor. With respect to such an adversary, we define the notion of IND-CCA security.
(3) Type-III adversary represents an outsider attacker, who obtains the trapdoor but does not own the server’s private key. With respect to such an adversary, we also define the notion of IND-CCA security.

(4) Type-IV adversary represents an outsider attacker, who does not own the receiver’s private key or the trapdoor. In fact, if our scheme can achieve IND-CCA security against a Type-II adversary and Type-III adversary, it is obvious that it also can achieve IND-CCA security against a Type-IV adversary since a Type-IV adversary gains less information than the Type-II adversary and the Type-III adversary during the attack game. Hence, we omit the security definition against a Type-IV adversary.

As to the trapdoor security of the PKE-DET scheme, since each receiver has only one unencrypted trapdoor, which is only related to the receiver’s private key without being related to any message, we define the notion of anonymous trapdoor.

In the following, we present the definitions with respect to these four privacy concerns.

**Definition 3.3 (OW-CCA Secure Against Type-I Adversary).** Let $\Pi = (\text{Setup}, \text{KeyGen}_s, \text{KeyGen}_r, \text{Encrypt}, \text{Decrypt}, \text{Delegate}, \text{Test})$ be a PKE-DET scheme and let $A$ be a polynomial-time (PPT) adversary. Let

$$\text{Adv}_{A,\text{Type-I}}^{\text{OW-CCA}} \equiv \Pr \left[ pp \leftarrow \text{Setup}(\lambda), (pk_r, sk_r) \leftarrow \text{KeyGen}_r(pp) \right.$$

$$\left. c^* \leftarrow \text{Encrypt}(pk_r, m^*), t^* \leftarrow \text{Delegate}(sk_r, pk_s) \right] - \frac{1}{2},$$

where

(1) $\mathcal{O}_D$: On input of a ciphertext $c$, it runs the Decrypt algorithm to return $m$ using the secret key $sk_r$.

(2) $\mathcal{O}_T$: On input of the server’s public key $pk_s$, it runs the Delegate algorithm to return $t$ using the secret key $sk_r$.

We restrict that $A$ should not query $\mathcal{O}_D$ for the decryption of $c^*$.

We say that $\Pi$ has OW-CCA secure against a Type-I adversary if $\text{Adv}_{A,\text{Type-I}}^{\text{OW-CCA}}$ is negligible for any $A$.

**Definition 3.4 (IND-CCA Secure Against Type-II Adversary).** Let $\Pi = (\text{Setup}, \text{KeyGen}_s, \text{KeyGen}_r, \text{Encrypt}, \text{Decrypt}, \text{Delegate}, \text{Test})$ be a PKE-DET scheme and let $A$ be a PPT adversary. Let

$$\text{Adv}_{A,\text{Type-II}}^{\text{IND-CCA}} \equiv \Pr \left[ pp \leftarrow \text{Setup}(\lambda), (pk_r, sk_r) \leftarrow \text{KeyGen}_r(pp) \right.$$

$$\left. (pk_s, sk_s) \leftarrow A^{\mathcal{O}_D}(pk_r, M), b \leftarrow \{0, 1\}, c^* \leftarrow \text{Encrypt}(pk_r, m_b) \right.$$

$$\left. b' \leftarrow A^{\mathcal{O}_T}(m_0, m_1, c^*): b' = b \right] - \frac{1}{2},$$

where $m_0 \neq m_1 \land |m_0| = |m_1|$ and oracle $\mathcal{O}_D$ is defined in the same way as in Definition 3.3. We say that $\Pi$ has IND-CCA (secure) against a Type-II adversary if $\text{Adv}_{A,\text{Type-II}}^{\text{IND-CCA}}$ is negligible for any $A$.

**Definition 3.5 (IND-CCA Secure Against Type-III Adversary).** Let $\Pi = (\text{Setup}, \text{KeyGen}_s, \text{KeyGen}_r, \text{Encrypt}, \text{Decrypt}, \text{Delegate}, \text{Test})$ be a PKE-DET scheme and let $A$ be a PPT adversary. Let

$$\text{Adv}_{A,\text{Type-III}}^{\text{IND-CCA}} \equiv \Pr \left[ pp \leftarrow \text{Setup}(\lambda), (pk_r, sk_r) \leftarrow \text{KeyGen}_r(pp) \right.$$

$$\left. (pk_s, sk_s) \leftarrow A^{\mathcal{O}_T}(pk_r, M), (m_0, m_1) \leftarrow A^{\mathcal{O}_D, \mathcal{O}_T}(pk_s, pk_r, M) \right.$$ 

$$\left. b \leftarrow \{0, 1\}, c^* \leftarrow \text{Encrypt}(pk_r, m_b) \right.$$ 

$$\left. t^* \leftarrow \text{Delegate}(sk_r, pk_s) \right.$$ 

$$\left. b' \leftarrow A^{\mathcal{O}_D, \mathcal{O}_T}(m_0, m_1, c^*, t^*): b' = b \right] - \frac{1}{2},$$

where $m_0 \neq m_1 \land |m_0| = |m_1|$ and oracle $\mathcal{O}_D, \mathcal{O}_T$ are defined in the same way as in Definition 3.3. We say that $\Pi$ has IND-CCA (secure) against a Type-III adversary if $\text{Adv}_{A,\text{Type-III}}^{\text{IND-CCA}}$ is negligible for any $A$. 

---

**FIGURE 2.** The overview of the PKE-DET scheme.
DEFINITION 3.6 (Anon-TD secure). Let \( \Pi = (\text{Setup, KeyGen}_S, \text{KeyGen}_R, \text{Encrypt, Decrypt, Delegate, Test}) \) be a PKE-DET scheme and let \( \mathcal{A} \) be a PPT adversary. Let

\[
\text{Adv}_{\mathcal{A}}^{\text{Anon-TD}} \overset{\text{def}}{=} \Pr \left[ pp \leftarrow \text{Setup}(\lambda), (pk_S, sk_S) \leftarrow \text{KeyGen}_S(pp) \mid (pk_R, sk_R) \leftarrow \mathcal{A}(pk_S), b \leftarrow (0, 1), \tau^* \leftarrow \text{Delegate}(sk_R, pk_S) \mid b^* \leftarrow \mathcal{A}(\tau^*) \ : b^* = b \right] - \frac{1}{2},
\]

where \( O_T \) is defined in the same way as in Definition 3.3. We say that \( \Pi \) has an anonymous trapdoor (Anon-TD secure) if \( \text{Adv}_{\mathcal{A}}^{\text{Anon-TD}} \) is negligible for any \( \mathcal{A} \).

Offline message recovery attack. Similarly to the offline keyword guessing attack in PEKS, the offline message recovery attack [25] exists in PKE-DET. This type of attack is unavoidable due to the desired functionality. For example, given a ciphertext \( c_A = \text{Encrypt}(m, pk_A) \) and a trapdoor \( t_A = \text{Delegate}(sk_A, pk_S) \), the server can test whether \( m' = m \) holds for any \( m' \in \mathcal{M} \) by checking the following equation:

\[
\text{Test}(c_A, t_A, \text{Encrypt}(m', pk_A), t_A, sk_S) = 1.
\]

Therefore, when the message space is polynomial size, the adversary delegated equality test on the ciphertexts is capable of mounting an offline message recovery attack by checking every \( m' \in \mathcal{M} \).

4. THE PROPOSED SCHEME

(1) \textbf{Setup}(\lambda): On input of the security parameter \( \lambda \) the algorithm outputs public parameters \( pp \) as follows.

(a) Generate Type 2 pairing parameters: group \( G_1, G_2, G_T \) of prime order \( p \), a bilinear map \( e : G_1 \times G_2 \rightarrow G_T \), generators of \( G_1: g_1, g_3 \), a generator of \( G_2: g_2 \).

(b) Select two hash functions: \( H_1 : G_T \rightarrow G_2 \) and \( H_2 : G_1 \times G_2 \rightarrow \{0, 1\}^{[k+\log(p)}, \) where \( k \) is a security parameter such that the elements of \( G_2 \) are represented in \( k \) bits.

(2) \textbf{KeyGen}_S(pp): On input the public parameters \( pp \) the algorithm selects \( \theta \in \mathbb{Z}_p^* \) and outputs the server’s key pair:

\[
(pk_S, sk_S) = (h = g_1^\theta, \theta).
\]

(3) \textbf{KeyGen}_R(pp): On input the public parameters \( pp \), the algorithm selects \( \alpha, \beta \in \mathbb{Z}_p^* \) and outputs the receivers’ key pair:

\[
(pk_R, sk_R) = ((u_1 = g_2^\alpha, u_2 = g_3^\beta), (\alpha, \beta)).
\]

(4) \textbf{Encrypt}(m, PK_R): It selects \( r_1, r_2 \in \mathbb{Z}_p \) and generates a ciphertext \( c = (c_1, c_2, c_3, c_4) \) for \( m \in \mathbb{G}_2^* \) (where \( \mathbb{G}_2 = \mathbb{G}_2(1/1) \)):

\[
c_1 = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m^{r_1} \cdot H_1(e(g_1, u_1)^2)
\]

\[
c_4 = H_2(c_1||c_2||c_3||u_2) \oplus m || r_1.
\]

(5) \textbf{Decrypt}(c, sk_R): It parses \( c \) as \( (c_1, c_2, c_3, c_4) \) and computes

\[
m || r_1 \leftarrow c_4 \oplus H_2(c_1||c_2||c_3||c_4^2),
\]

and outputs \( m \) if the following equations hold:

\[
c_1 = g_1^{r_1}, \quad c_3 = m^{r_1} \cdot H_1(e(g_1, c_2)^2).
\]

If either equation does not hold, the algorithm outputs an error symbol \( \perp \).

(6) \textbf{Delegate}(sk_R, pk_S): It selects \( r \in \mathbb{Z}_p \) and generates a trapdoor to support the equality test on the encrypted data:

\[
t = (t_1, t_2) = (g_3^r, g_1^{r^2}, h^{r^3}).
\]

Suppose that the server obtains the receiver \( R \)'s trapdoor \( t_R = (t_{R,1}, t_{R,2}) \) and \( R' \)'s trapdoor \( t_{R'} = (t_{R',1}, t_{R',2}) \), then it can run the following test algorithm to perform the equality test on \( R' \)'s ciphertext \( c' \) and \( R \)'s ciphertext \( c \).

(7) \textbf{Test}(c, c', t_R, t_{R'}, sk_R): It performs as follows:

(a) Parse \( c \) as \( (c_1, c_2, c_3, c_4), c' \) as \( (c'_1, c'_2, c'_3, c'_4) \), \( t_R \) as \( (t_{R,1}, t_{R,2}) \) and \( t_{R'} \) as \( (t_{R',1}, t_{R',2}) \).

(b) Compute \( t_3 = t_{R,2}/(t_{R,1})^0 \) and \( t_4 = t_{R',2}/(t_{R',1})^0 \).

(c) Compute \( X \) using \( t_3 \) and \( X' \) using \( t_4 \):

\[
X = H_1(e(t_3, c_2)), \quad X' = H_1(e(t_4, c'_2)).
\]

(d) Output 1 if the following equation holds, and output 0 otherwise:

\[
e(c_1, X') = e(c'_1, X).
\]

THEOREM 4.1. The above PKE-DET scheme is sound according to Definition 3.2.

Proof. We show that three conditions in Definition 3.2 are satisfied.

(1) As to the first condition, given \( (pk_R, sk_R) \leftarrow \text{KeyGen}(pp), c = (c_1, c_2, c_3, c_4) \leftarrow \text{Encrypt}(pk_R, m) \), since \( c_2 = (g_2^r)^2 = (g_3^r)^2 = u_2^{r_2} \), we have

\[
c_4 \oplus H_2(c_1||c_2||c_3||u_2) = c_4 \oplus H_2(c_1||c_2||c_3||u_2^2) = m || r_1.
\]

And then it is easy to verify that \( c_1 = g_1^{r_1}, c_3 = m^{r_1} \cdot H_1(e(g_1, c_2)^2) \), which implies that \( \text{Decrypt(Encrypt}(m, pk_R), sk_R) = m \).
5. SECURITY ANALYSIS

Theorem 5.1. The above PKE-DET scheme is OW-CCA secure against a Type-1 adversary according to Definition 3.3 in the random oracle model under the CDH and co-DH assumptions.

Proof. Let $A$ be a PPT adversary attacking the OW-CCA security against a Type-I adversary. Suppose that $A$ runs in time $rt$ and makes at most $q_{HK}$ hash queries, $q_{DC}$ decryption queries and $q_{T}$ trapdoor queries. Let $\text{Adv}^{\text{OW-CCA}}_{\text{Type-I}}(rt, q_{HK}, q_{DC}, q_{T})$ denote the advantage of $A$ in Definition 3.3. The security proof is done through a sequence of games.

Game 0: We define Game 0 to be the attack game against $A$ in Definition 3.3.

1. $\alpha, \beta \leftarrow \mathbb{Z}_p^*$, $u_1 = g_2^\alpha$, $u_2 = g_2^\beta$.
2. $(h = g_3^\alpha, \theta \leftarrow \mathbb{Z}_p^*) \leftarrow \mathcal{A}^{\text{OW-CCA}}(\alpha, \beta)$.

Game 1:

1. $\alpha, \beta \leftarrow \mathbb{Z}_p^*$, $u_1 = g_2^\alpha$, $u_2 = g_2^\beta$, $T_2 = \emptyset$.
2. $(h = g_3^\alpha, \theta \leftarrow \mathbb{Z}_p^*) \leftarrow \mathcal{A}^{\text{OW-CCA}}(\alpha, \beta)$.

We define $S_0$ to be the event that $m' = m$ in Game 0. It should be evident that this algorithm faithfully represents the attack game, so

$$\text{Adv}^{\text{OW-CCA}}_{\text{Type-I}}(rt, q_{HK}, q_{DC}, q_{T}) = \Pr[S_0]. \quad (1)$$

Next we modify Game 0 and obtain the following game.

Game 1:

1. $\alpha, \beta \leftarrow \mathbb{Z}_p^*$, $u_1 = g_2^\alpha$, $u_2 = g_2^\beta$, $T_2 = \emptyset$.
2. $(h = g_3^\alpha, \theta \leftarrow \mathbb{Z}_p^*) \leftarrow \mathcal{A}^{\text{OW-CCA}}(\alpha, \beta)$.

$\alpha, \beta$ are uniformly selected as above. $u_1$ and $u_2$ are also uniformly selected. Therefore, the above game has the same distribution of its inputs as Game 0. We have

$$\Pr[S_0] = \Pr[S_1] = \frac{1}{|\mathbb{Z}_p^*|}.$$
We claim that the probability if the CDH problem is intractable.

Let $S_1$ be the event that $m' = m$ in Game 1. Owing to the idealness of the random oracle, Game 1 is identical to Game 0. We claim that

$$\Pr[S_0] = \Pr[S_1].$$  \hspace{1cm} (2)

In the next game, we further modify the simulation in an indistinguishable way.

**Game 2:**

1. $(\alpha, \beta) \leftarrow Z_p^*, u_1 = g_1^{\alpha}, u_2 = g_2^{\beta}, T_2 = \emptyset.$
2. $(h = g_1^{\alpha}, \theta_R \in Z_p^*) \leftarrow A^{C_{\alpha}, C_{\beta}}(u_1, u_2)$.
3. $m \leftarrow G_2^*, r_1, r_2 \leftarrow Z_p^*, W_2^* \leftarrow [0, 1]^{k + \log p}, c^* = (c_1^*, c_2^*, c_3^*, c_4^*)$ defined as follows:
   $$c_1^* = g_1^{\alpha}, \quad c_2^* = g_2^{\beta}, \quad c_3^* = m^{r_1} \cdot H_1(e(g_1, u_1)^{T_2}), \quad c_4^* = W_2^*.$$
   And add the tuple $(c_1^*, c_2^*, c_3^*, (c_4^*)^\beta, W_2^* \oplus (m||r_1))$ to the table $T_2$ for $H_2$.
4. $r^* \leftarrow \text{Delegate}(\alpha, h)$.
5. $m' \leftarrow A^{C_{\alpha}, C_{\beta}, C_{r^*}}(r^*, c^*)$, where the oracles work as follows.

**H$_2$ oracle query:** It is simulated in the same way as that in Game 1 except that if $A$ asks $(\cdot, c_1^*, \cdot, (c_2^*)^\beta)$, the game is aborted. Let this event be $E_1$.

**$O_D$ oracle query:** The same as that in Game 1 except if $A$ asks for decryption of $(c_1^*, c_2^*, c_3^*$, $c_4^*)$ where $c_4^* \neq (c_4^*)^2$, it is returned.

**$O_T$ oracle query:** It is simulated in the same way as that in Game 1.

Let $S_2$ be the event that $m' = m$ in Game 2. The challenge ciphertext generated in this game is identically distributed to that in Game 1, as $c_3^*$ is a random value in both Games 1 and 2. Therefore, if $E_1$ does not occur, Game 2 is identical to Game 1. We claim that

$$\Pr[S_2] = \Pr[S_1].$$  \hspace{1cm} (3)

Next, we show that the event $E_1$ occurs with negligible probability:

$$\Pr[E_1] \leq \text{Adv}^{\text{CDH}} \frac{q_D}{2^{k + \log p}}. \hspace{1cm} (4)$$

**Lemma 5.1.** Event $E_1$ happens in Game 2 with negligible probability if the CDH problem is intractable.

**Proof.** Suppose that $\Pr[E_1]$ is non-negligible. We construct a PPT algorithm $B$ to break the CDH assumption. Given a tuple $(g, g^a, g^b) \in G_2^2$, $B$ selects $(\alpha, x, r_1) \in (Z_p^*)^3$, $g_1 \in G_1$ and $W_2^* \in [0, 1]^{k + \log p}$, and then sets the public parameters:

$$g_1, g_2 = g, \quad u_1 = g^a, \quad u_2 = g^b.$$

We see that $B$ invokes $A$ by running the $\text{Delegate}$ algorithm to obtain $g_3^a$ using $(h, \theta)$ owned by $A$. And then $B$ can generate the challenge ciphertext $c^* = (c_1^*, c_2^*, c_3^*, c_4^*)$ for $m = g^a$, which is defined as follows:

$$c_1^* = g_1^\alpha, \quad c_2^* = g_2^\beta, \quad c_3^* = g^{xa} \cdot H_1(e(g_1, g_2)^{T_2}), \quad c_4^* = W_2^*.$$

It then adds $(c_1^*, c_2^*, c_3^*, c_4^*)$ to table $T_2$, which is initially empty, where $T$ represents that the value is unknown; $B$ checks if $e(g_2, Z) = e(u_2, c_2^*)$. If the equation holds, $B$ outputs $Z$ and aborts the game.

**H$_2$ oracle query:** $B$ simulates the oracle as described in Game 2 except that if $A$ makes a query on $(\cdot, c_1^*), (\cdot, Z)$, $B$ checks if the challenge ciphertext $e^*$ with the same distribution as that in Game 1. The oracles for $A$ are simulated as follows:

$$\Pr[D_1] = \text{Adv}^{\text{CDH}}. \hspace{1cm} (5)$$

If $D_1$ does not occur, $B$ aborts with failure.

Next, we analyze Game 2 and the simulation for the event $D_1$ are indistinguishable. It is evident that $H_2$ oracle queries can be simulated perfectly. We only focus on the decryption queries:

1. $(c_1^*, c_2^*, c_3^*, c_4^*)$ has been queried to $H_2$ oracle before a decryption query $(c_1, c_2, c_3, c_4)$ is issued. In this case, $c_4$ is uniquely determined after $(c_1, c_2, c_3, c_4)$ is queried to $H_2$ oracle. So the decryption oracle is simulated perfectly.

2. $(c_1, c_2, c_3, c_4)$ has never been queried to $H_2$ oracle before a decryption query $(c_1, c_2, c_3, c_4)$ is issued. In this case, $\perp$ is returned by the decryption oracle. The
We see that \((g, gb)\) is a valid ciphertext. However, due to the idealness of the random oracle, this probability is \(1/2^{k+\log p}\).

Denote by \(D_1\) the event that a valid ciphertext is rejected in the simulation; then we have
\[
\Pr[D_1] \leq \frac{q_D}{2^{k+\log p}}.
\]
(6)

Therefore, if event \(D_1\) does not happen, then the simulation is identical to \(Game 2\), and we claim that
\[
|\Pr[D_1] - \Pr[E_1]| \leq \Pr[D_2].
\]
(7)
Owing to Equations (5)–(7), we have
\[
\text{Adv}^\text{CDH}_B \geq \Pr[E_1] - \frac{q_D}{2^{k+\log p}}.
\]
(8)

So if \(\Pr[E_1]\) is non-negligible, then the probability of breaking the CDH assumption is non-negligible. This completes the proof of Lemma 5.1.

Then we show that event \(S_2\) occurs with negligible probability:
\[
\Pr[S_2] \leq \text{Adv}^\text{co-DH}.
\]
(9)

**Lemma 5.2.** Event \(S_2\) happens in \(Game 2\) with negligible probability if the co-DH problem is intractable.

**Proof.** Suppose that \(\Pr[S_2]\) is non-negligible. We construct a PPT algorithm \(B\) to break the co-DH assumption. Given a tuple \((g, g^h) \in G^*_2\) and \((g', g'^h) \in G^*_2\) where \(g = \psi(g')\), \(B\)'s goal is to compute \(g^{bc}\); \(B\) selects \((\alpha, \beta, r_2) \in R(\mathbb{Z}_p^\ast)^3, g_2 \in G_2\) and \(W_2 \in \{0, 1\}^k+\log p\), and then sets the public parameters:
\[
g_1 = g^h, \quad g_2 = g_2^\ast, \quad u_1 = g_2^\ast, \quad u_2 = g_2^\beta.
\]
We see that \(B\) invokes \(A\) by running the Delegate algorithm to obtain \(g_2^\ast\) using \((h, \theta)\) owned by \(A\). Then it generates the challenge ciphertext \(c^* = (c_1^\ast, c_2^\ast, c_2'\ast)\) for \(m^\ast = g^{vc}\), which is defined as follows:
\[
c_1^\ast = g, \quad c_2^\ast = g_2^\ast, \quad c_3^\ast = g^{vc} \cdot H_1(e(g_1, u_1)^{c_2^\ast}),
\]
\[
c_4^\ast = W_2^\ast.
\]
It then adds \((c_1^\ast, c_2^\ast, c_3^\ast, (c_2\beta, \top))\) into table \(T_2\) which is initially empty, where \(\top\) represents that the value is unknown. We see that \(B\) invokes adversary \(A\) on input \(PKR = (u_1, u_2)\) and the ciphertext \(c^*\) with the same distribution as that in \(Game 2\). \(B\) simulates the game by the description of \(Game 2\). Finally, \(B\) outputs whatever \(A\) outputs. We claim that, given
\[
\left(g : g_1^\ast, \quad g^h : g_1, \quad g^{vc} : \frac{c_3}{H_1(e(g_1, u_1)^{c_2^\ast})} = m^\ast\right),
\]
\(B\) obtains \(m = g^{bc}\) by invoking \(A\). Therefore,
\[
\Pr[S_2] \leq \text{Adv}^\text{co-DH}.
\]

So if \(\Pr[S_2]\) is non-negligible, then the probability of breaking the co-DH assumption is non-negligible. This completes the proof of Lemma 5.2.

Therefore, due to Equations (1)–(4) and (9), we claim that
\[
\text{Adv}^\text{IND-CCA}_A(\sigma_1, q_{H_1}, q_D, q_T) \leq \text{Adv}^\text{CDH} + \text{Adv}^\text{co-DH} + \frac{q_D}{2^{k+\log p}}.
\]
(10)

This completes the proof of Theorem 5.1.

**Theorem 5.2.** The above PKE-DET scheme is IND-CCA secure against a Type-II adversary according to Definition 3.4 in the random oracle model under the CDH and the BDH assumptions.

**Proof.** Let \(A\) be a PPT adversary attacking IND-CCA security against a Type-II adversary. Suppose that \(A\) runs in time \(rt\) and makes at most \(q_{H_1}\) hash queries, \(q_D\) hash queries, \(q_T\) decryption queries and \(q_T\) trapdoor queries. Let \(\text{Adv}^\text{IND-CCA}_A(rt, q_{H_1}, q_D, q_T)\) denote the advantage of \(A\) in Definition 3.4. The security proof is done through a sequence of games.

**Game 0:** We define \(Game 0\) to be the attack game against \(A\) in Definition 3.4.

(1) \(\alpha, \beta \leftarrow Z_p^\ast, u_1 = g_1^\alpha, u_2 = g_2^\beta.
\)
(2) \(\theta \in R (Z_p^\ast), \quad h = g_1^\theta \in \mathbb{G}_1^*, (m_0, m_1) \in R (\mathbb{G}_2^2) \leftarrow A^{\alpha, \beta, c_{\theta, 0}, c_{\theta, 1}}(u_1, u_2).
\)
(3) \(b \leftarrow \{0, 1\}, \quad r_1, r_2 \leftarrow Z_p^\ast, \quad c^* = (c_1^*, c_2^*, c_3^*, c_4^*)\) defined as follows:
\[
c_1^* = g_1^\alpha, \quad c_2^* = g_2^\beta, \quad c_3^* = m_0^\beta \cdot H_1(e(g_1, u_1)^{c_2^*}), \quad c_4^* = H_2(c_1^*e_2^*) \oplus m_1||r_1.
\]
(4) \(b' \in \{0, 1\} \leftarrow A^{\alpha, \beta, c_{\theta, 0}, c_{\theta, 1}}(m_0, m_1, c^*).
\)

\(H_1\) oracle query: On input \(\tau \in \mathbb{G}_T\), a random value \(h_1 \in \mathbb{G}_2\) is returned, and meanwhile if the same input is asked multiple times, the same answer will be returned.

\(H_2\) oracle and \(OD\) oracle query: They are simulated in the same way as that in \(Game 0\) in the proof of Theorem 5.1.

We define \(F_0\) be the event that \(b' = b\) in \(Game 0\). It should be evident that this algorithm faithfully represents the attack game, so
\[
\text{Adv}^\text{IND-CCA}_A(rt, q_{H_1}, q_D, q_T) = |\Pr[F_0] - \frac{1}{2}|.
\]
(11)

Next we modify \(Game 0\) and obtain the following game.

**Game 1:**

(1) \(\alpha, \beta \leftarrow Z_p^\ast, u_1 = g_1^\alpha, u_2 = g_2^\beta, \quad T_2 = \emptyset.
\)
(2) \(\theta \in R (Z_p^\ast), \quad h = g_1^\theta \in \mathbb{G}_1^*, (m_0, m_1) \in R (\mathbb{G}_2^2) \leftarrow A^{\alpha, \beta, c_{\theta, 0}, c_{\theta, 1}}(u_1, u_2).
\)
We claim that in ciphertext generated in this game is identically distributed to $(\alpha, \beta)$ defined as follows:

\[
c_i = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m_0^{r_1} \cdot H_1(e(g_1, u_2)^{r_2}),
\]

\[
c_4 = U_2^* \oplus (m_0 || r_1).
\]

And add the tuple $(c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ to the table $T_2$ for $H_2$.

(4) $b' \in \{0, 1\}$, $r_1, r_2 \gets \mathbb{Z}_p^*, U^* \gets [0, 1]^{k + \log p}, c^* = (c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ defined as follows:

\[
c_i = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m_0^{r_1} \cdot H_1(e(g_1, u_2)^{r_2}),
\]

\[
c_4 = U_2^*.
\]

And add the tuple $(c_1, c_2, c_3, (c_4)^\beta, U_2^* \oplus (m || r_1))$ to the table $T_2$ for $H_2$.

(4) $b' \in \{0, 1\}$, $r_1, r_2 \gets \mathbb{Z}_p^*, U^* \gets [0, 1]^{k + \log p}, c^* = (c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ defined as follows:

\[
c_i = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m_0^{r_1} \cdot H_1(e(g_1, u_2)^{r_2}),
\]

\[
c_4 = U_2^*.
\]

Let $F_0$ be the event that $b' = b$ in Game 1. Owing to the idealness of the random oracle, $Game 1$ is identical to $Game 0$. We claim that

\[
Pr[F_0] = Pr[F_1].
\]

In the next game, we further modify the simulation in an indistinguishable way.

**Game 2:**

1. $\alpha, \beta \leftarrow \mathbb{Z}_p^*, u_1 = g_2^\alpha, u_2 = g_2^\beta, T_2 = \emptyset$.
2. $(\theta \in \mathbb{Z}_p^*, h = g_2^\beta \in \mathbb{G}_2^*, (m_0, m_1) \in \mathbb{G}_2^2) \leftarrow A_{O_H, O_C, O_D}(u_1, u_2)$.
3. $b \leftarrow \{0, 1\}, r_1, r_2 \leftarrow \mathbb{Z}_p^*, U^* \leftarrow [0, 1]^{k + \log p}, c^* = (c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ defined as follows:

\[
c_i = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m_0^{r_1} \cdot H_1(e(g_1, u_2)^{r_2}),
\]

\[
c_4 = U_2^*.
\]

And add the tuple $(c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ to the table $T_2$ for $H_2$.

(4) $b' \in \{0, 1\}$, $r_1, r_2 \gets \mathbb{Z}_p^*, U^* \gets [0, 1]^{k + \log p}, c^* = (c_1, c_2, c_3, (c_4)^\beta, U_2^*)$ defined as follows:

\[
c_i = g_1^{r_1}, \quad c_2 = g_2^{r_2}, \quad c_3 = m_0^{r_1} \cdot H_1(e(g_1, u_2)^{r_2}),
\]

\[
c_4 = U_2^*.
\]

Let $F_2$ be the event that $b' = b$ in Game 2. The challenge ciphertext generated in this game is identically distributed to that in Game 1, as $c_3$ is a random value in both Games 1 and 2. Therefore, if $E_0$ does not occur, Game 2 is identical to Game 1. We claim that

\[
Pr[F_2] - Pr[F_1] \leq Pr[E_1].
\]

Similar to the proof of Lemma 5.1, we can prove the following claim:

\[
Pr[E_1] \leq Adv^{CDH} + \frac{q_D}{2^{k + \log p}}.
\]

In the next game, we further modify the simulation in an indistinguishable way.
Event $D_3$ denotes that a valid ciphertext is rejected in the
decryption. We claim that

$$\Pr[D_3] \leq \frac{qd}{2^k}. \quad (15)$$

If $D_1$ does not happen, Game 3 is identical to Game 2, so we have

$$\Pr[F_3] - \Pr[F_2] \leq \Pr[D_3]. \quad (16)$$

Owing to Equations (15) and (16), we have

$$\Pr[F_2] - \Pr[F_1] \leq \frac{qd}{2^k}. \quad (17)$$

In the next game, we further modify the simulation in an
indistinguishable way.

**Game 4:**

1. $\alpha, \beta \leftarrow \mathcal{Z}_p^k$, $u_1 = g_2^\alpha$, $u_2 = g_2^\beta$, $T_1 = \emptyset$, $T_2 = \emptyset$.
2. $\{\theta_R \in \mathcal{Z}_p^k, h = g_2^\theta \in G_1^\ast, (m_0, m_1) \in \mathcal{E}(G_2^\ast)^2\} \leftarrow \mathcal{A}^{\theta_R, c_{\theta_R}, c_{\theta_R}(u_1, u_2)}$.
3. $b \leftarrow \{0, 1\}$, $r_1, r_2 \leftarrow \mathcal{Z}_p$, $U_{b, 1} \leftarrow \{0, 1\}^k$, $U_{b, 2} \leftarrow \{0, 1\}^{k + \log p}$, $c^\ast = (c_1^\ast, c_2^\ast, c_3^\ast)$ defined as follows:

$$c_1^\ast = g_1^r_1, \quad c_2^\ast = g_2^r_2, \quad c_3^\ast = U_{b, 1}^\ast,$$

$$c_4^\ast = U_{b, 2}^\ast.$$

And add the tuple $(c_1^\ast, c_2^\ast, c_3^\ast, (c_2^e)^\ast, U_{b, 2}^\ast \oplus (m_b||r_1))$ to the table $T_2$ for $H_2$, and the tuple $(e(g_1, u_1)^{r_1}, U_{b, 1}^\ast/m_{b}^\ast)$ to the table $T_1$ for $H_1$.

4. $b^\prime \leftarrow \{0, 1\}$, $r_3 \leftarrow \mathcal{A}^{\theta_R, c_{\theta_R}, c_{\theta_R}(m_0, m_1, c^\ast)}$.

**$H_1$ oracle query:** It is simulated in the same way as that in Game 3 except that if $A$ asks $e(g_1, c_2^e)^\ast$, the game is aborted. Let this event be $E_2$.

**$H_2$ oracle and $O_D$ oracle:** They are simulated in the same way as that in Game 3.

First, we analyze when event $E_2$ happens, $A$ can guess correctly the value of $b$ with probability 1. $A$ computes $m_0 = g_2^{a_0}$ and $m_1 = g_2^{a_1}$ for $(a_0, a_1) \in \mathcal{E}(Z_p)^2$ and sends $(m_0, m_1)$ to the challenger. After receiving the challenge ciphertext $(c_1^\ast, c_2^\ast, c_3^\ast, c_4^\ast)$, $A$ checks if $e(g_1, c_3^\ast) = h_1(e(g, c_2^e)^\ast)$. If yes, $A$ returns 0; otherwise, $A$ returns 1. So $A$ can guess the value of $b$ correctly.

The challenge ciphertext generated in this game is identically distributed to that in Game 3, as $c_3$ is a random value in both Games 3 and 4. Therefore, if $E_2$ does not occur, Game 4 is identical to Game 3. We claim that

$$\Pr[F_2] - \Pr[F_1] \leq \Pr[E_2]. \quad (18)$$

Next, we first show that event $E_2$ occurs with negligible probability:

$$\Pr[E_2] \leq q_{H_1} \text{Adv}^{BDH} + \frac{qd}{2^k}. \quad (19)$$

**Lemma 5.3.** Event $E_2$ happens in Game 4 with negligible probability if the BDH problem is intractable.

**Proof.** Suppose that $\Pr[E_2]$ is non-negligible. We construct a PPT algorithm $B$ to break the BDH assumption. Given a tuple $(g, g^\ast) \in G_1^\ast$ and $(g', g^\gamma, g'^\gamma) \in G_1^\ast$, where $g = \psi(g')$, $B$'s goal is to compute $e(g, g^\gamma)^\ast$; $B$ selects $(\beta, r_1) \in \mathcal{E}(Z_p)^2$, $U_{b, 1}^\ast \in \{0, 1\}^k$ and $U_{b, 2}^\ast \in \{0, 1\}^{k + \log p}$, and sets the public parameters:

$$g_1 = g^\ast, \quad g_2 = g', \quad u_1 = g^{\gamma}, \quad u_2 = g_2^\beta,$$

and generates the challenge ciphertext $c^\ast = (c_1^\ast, c_2^\ast, c_3^\ast, c_4^\ast)$, which is defined as follows:

$$c_1^\ast = (g^\ast)^{r_1}, \quad c_2^\ast = g_2^{r'}, \quad c_3^\ast = U_{b, 1}^\ast; \quad c_4^\ast = U_{b, 2}^\ast.$$

It then adds $(c_1^\ast, c_2^\ast, c_3^\ast, (c_2^e)^\ast, T)$ to table $T_2$; $T_1$ is initially empty. We see that $B$ invokes adversary $A$ on input $PK = (u_1, u_2)$ and the ciphertext $c^\ast$ with the same distribution as that in Game 4; $B$ simulates the game by the description of Game 4. The oracles for $A$ are simulated as follows.

**$H_1$ oracle query:** They are simulated in the same way as that in Game 4.

**$O_D$ oracle query:** On input $(c_1, c_2, c_3, c_4)$, if $A$ asks for decryption of $(c_1^\ast, c_2^\ast, c_3^\ast, c_4^\ast)$ where $(c_4^\ast) \neq c_4^\ast$, the response is $\perp$; otherwise, it returns $c_4^\ast$.

Next, we analyze the decryption queries:

1. $e(g_1, c_2)^\ast$ has been queried to $H_1$ oracle before a decryption query $(c_1, c_2, c_3, c_4)$ is issued. In this case, $U_{b, 1}^\ast$ is uniquely determined after $e(g_1, c_2)^\ast$ is queried to $H_1$ oracle. So the decryption oracle is simulated perfectly.

2. $e(g_1, c_2)^\ast$ has never been queried to $H_1$ oracle before a decryption query $(c_1, c_2, c_3, c_4)$ is issued. In this case, $\perp$ is returned by the decryption oracle. The simulation fails if $(c_1, c_2, c_3, c_4)$ is a valid ciphertext. However, due to the indelibility of the random oracle, this happens with probability $1/2^k$.

Denote by $D_4$ the event that a valid ciphertext is rejected in the simulation; then we have

$$\Pr[D_4] \leq \frac{qd}{2^k}. \quad (20)$$

Therefore, if event $D_4$ does not happen and $(e(g_1, c_2)^\ast, \perp)$ appears in some tuple on the $H_1$ table with probability at least $\Pr[E_2]$, $B$ obtains $e(g, g^\gamma)^\ast$ denoted by the solution of the
BDH problem. This means that \( \Pr[E_2|\neg D_4] = (1/q_H)\Pr[E_2] \); otherwise, \( B \) aborts with failure. Therefore, we have

\[
\Pr[E_2] = \Pr[E_2|D_4]\Pr[D_4] + \Pr[E_2|\neg D_4]\Pr[\neg D_4] \\
\geq \Pr[E_2|\neg D_4]\Pr[\neg D_4] \\
= \frac{1}{q_H}\Pr[E_2]\Pr[\neg D_4] \\
= \frac{1}{q_H}(\Pr[E_2](1 - \Pr[D_4])) \\
\geq \frac{1}{q_H}(\Pr[E_2] - \Pr[D_4]) \\
\geq \frac{1}{q_H}\left(\Pr[E_2] - \frac{q_D}{2^2}\right). \quad (21)
\]

Therefore,

\[
\text{Adv}^{\text{BDH}}_B \geq \frac{1}{q_H}\left(\Pr[E_2] - \frac{q_D}{2^2}\right). \quad (22)
\]

So if \( \Pr[E_2] \) is non-negligible, the probability of breaking the BDH assumption is non-negligible. This completes the proof of Lemma 5.3.

**Lemma 5.4.** Event \( F_4 \) happens in Game 4 with probability \( \frac{1}{2} \).

\[
\Pr[F_4] = \frac{1}{2}. \quad (23)
\]

**Proof.** It is evident that all items in the challenge ciphertext are independent of the message \( m \), and so \( A \) can guess \( b' = b \) successfully without any advantage. Lemma 5.4 now follows.

Owing to Equations (11), (12)–(14), (17)–(19) and (23), we claim that

\[
\text{Adv}^{\text{IND-CCA}_{A,\text{Type-II}}}(rt, q_H, q_{H_1}, q_{D}, q_{D'}) \leq \text{Adv}^{\text{CDH}} + \frac{q_D}{2^{k-1}} + q_H\text{Adv}^{\text{BDH}}. \quad (24)
\]

This completes the proof of Theorem 5.2.

**Theorem 5.3.** The above PKE-DET scheme is IND-CCA secure against Type-III adversary according to Definition 3.5 in the random oracle model under the CDH, BDH and DDH assumptions.

**Proof.** The proof of Theorem 5.3 is similar to the proof of Theorem 5.2, where the difference is that the adversary can obtain the trapdoor of the challenge ciphertext. However, since the adversary does not obtain useful information from the trapdoor without the server’s private key, the ciphertext security of the PKE-DET scheme can still achieve IND-CCA security. Next we will give a simplified description to show the differences between the proof of Theorem 5.2 and the proof of Theorem 5.3. For example, the description of \( H_1 \) oracle, \( H_2 \) oracle and \( O_D \) oracle queries is omitted because they are simulated in the same way as that in Game \( i \) \((i = 0, 1, 2, 3, 4)\) in the proof of Theorem 5.2.

**Game 0:**

\[
(1) \alpha, \beta, \theta \leftarrow \mathcal{Z}_p^*, u_1 = g_2^u, u_2 = g_2^\beta, h = g_2^\theta.
\]

\[
(2) (m_0, m_1) \leftarrow (\mathcal{G}_2^*)^2 \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1, \gamma)(u_1, u_2, h).
\]

\[
(3) b \leftarrow \{0, 1\}, r_1, r_2, r_3 \leftarrow \mathcal{Z}_p^*, c^* = (c_1^*, c_2^*, c_3^*, c_4^*) \text{ is defined the same as that in the proof of Theorem 5.2 and } t^* \text{ is defined as } t^* = (g_3^c, g_3^t, h^s).
\]

\[
(4) b' \in \{0, 1\} \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1)(m_0, m_1, c^*, t^*).
\]

**\( O_T \) oracle query:** On input the server’s public key \( h \), it returns a trapdoor \( t \) by computing \( t = (g_3^c, g_3^t, h^s) \).

We define \( Q_3 \) to be the event that \( b' = b \) in Game 0. It should be evident that this algorithm faithfully represents the attack game, and so

\[
\text{Adv}^{\text{IND-CCA}_{A,\text{Type-II}}}(rt, q_H, q_{H_1}, q_{D}, q_{D'}) = |\Pr[Q_3] - \frac{1}{2}|. \quad (25)
\]

**Game 1:**

\[
(1) \alpha, \beta, \theta \leftarrow \mathcal{Z}_p^*, u_1 = g_2^u, u_2 = g_2^\beta, h = g_2^\theta, T_2 = \emptyset.
\]

\[
(2) (m_0, m_1) \leftarrow (\mathcal{G}_2^*)^2 \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1)(u_1, u_2, h).
\]

\[
(3) b \leftarrow \{0, 1\}, r_1, r_2, r_3 \leftarrow \mathcal{Z}_p^*, U_1^* \leftarrow \{0, 1\}^{k+\log p}, c^* = (c_1^*, c_2^*, c_3^*, c_4^*) \text{ is defined the same as that in the proof of Theorem 5.2 and } t^* \text{ is defined as } t^* = (g_3^c, g_3^t, h^s).
\]

\[
(4) b' \leftarrow \{0, 1\} \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1)(m_0, m_1, c^*, t^*).
\]

**\( O_T \) oracle query:** It is simulated in the same way as that in Game 0.

Let \( Q_1 \) be the event that \( b' = b \) in Game 1. Owing to the idealness of the random oracle, we claim that

\[
\Pr[Q_0] = \Pr[Q_1]. \quad (26)
\]

**Game 2:**

\[
(1) \alpha, \beta, \theta \leftarrow \mathcal{Z}_p^*, u_1 = g_2^u, u_2 = g_2^\beta, h = g_2^\theta, T_2 = \emptyset.
\]

\[
(2) (m_0, m_1) \leftarrow \mathcal{R} (\mathcal{G}_2^*)^2 \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1)(u_1, u_2, h).
\]

\[
(3) b \leftarrow \{0, 1\}, r_1, r_2, r_3 \leftarrow \mathcal{Z}_p^*, U_1^* \leftarrow \{0, 1\}^{k+\log p}, c^* = (c_1^*, c_2^*, c_3^*, c_4^*) \text{ is defined the same as that in the proof of Theorem 5.2 and } t^* \text{ is defined as } t^* = (g_3^c, g_3^t, h^s).
\]

\[
(4) b' \in \{0, 1\} \leftarrow A^{\text{Oh}_{H_1}}(c_0, c_1, c_0, c_1)(m_0, m_1, c^*, t^*).
\]

**\( O_T \) oracle query:** It is simulated in the same way as that in Game 1.

Let \( Q_2 \) be the event that \( b' = b \) in Game 2. Similar to Equations (13) and (14), we claim that

\[
|\Pr[Q_2] - \Pr[Q_1]| \leq \text{Adv}^{\text{CDH}} + \frac{q_D}{2^{k-1}q_H}. \quad (27)
\]
Game 3:

1. \( \alpha, \beta, \theta \leftarrow \mathbb{Z}_p^*, u_1 = g_2^\alpha, u_2 = g_2^\beta, h = g_3^{\theta}, T_1 = \emptyset, T_2 = \emptyset. \)
2. \((m_0, m_1) \in_R \mathbb{G}_2^2 \leftarrow \mathcal{A}_{\text{OM}, \text{CH}, \text{CQ}, \text{Cf}}(u_1, u_2, h).\)
3. \(b \leftarrow [0, 1], r_1, r_2, r_3 \leftarrow \mathbb{Z}_p^*, U_{3,1}^* \leftarrow \mathcal{G}_2, U_{3,2}^* \leftarrow [0, 1]\cdot^k \log p, \epsilon^* = (c_1^*, c_2^*, c_3^*, c_4^*) \) is defined the same as that in the proof of Theorem 5.2 and \( t^* = (g_3^{\epsilon_1^*}, g_3^{\epsilon_2^*} \cdot h^{r_1}). \)
4. \(b' \in [0, 1] \leftarrow \mathcal{A}_{\text{OM}, \text{CH}, \text{CQ}, \text{Cf}}(m_0, m_1, \epsilon^*, t^*). \)

\( \mathcal{O}_T \) oracle query: It is simulated in the same way as that in Game 2.

Let \( Q_3 \) be the event that \( b' = b \) in Game 3. Similar to Equation (17), we claim that

\[
|\Pr[Q_3] - \Pr[Q_2]| \leq \frac{q_D}{2^k}. \tag{28}
\]

Game 4:

1. \( \alpha, \beta, \theta \leftarrow \mathbb{Z}_p^*, u_1 = g_2^\alpha, u_2 = g_2^\beta, h = g_3^{\theta}, T_1 = \emptyset, T_2 = \emptyset. \)
2. \((m_0, m_1) \in_R \mathbb{G}_2^2 \leftarrow \mathcal{A}_{\text{OM}, \text{CH}, \text{CQ}, \text{Cf}}(u_1, u_2, h).\)
3. \(b \in_R [0, 1], r_1, r_2, r_3 \in \mathbb{Z}_p^*, U_{3,1}^* \leftarrow \mathbb{G}_2, U_{3,2}^* \leftarrow \mathbb{G}_2, U_{3,2}^* \leftarrow [0, 1]^{k \cdot \log p}, \epsilon^* = (c_1^*, c_2^*, c_3^*, c_4^*) \) is defined the same as that in the proof of Theorem 5.2 and \( t^* = (g_3^{\epsilon_1^*}, g_3^{\epsilon_2^*} \cdot h^{r_1}). \)
4. \(b' \in [0, 1] \leftarrow \mathcal{A}_{\text{OM}, \text{CH}, \text{CQ}, \text{Cf}}(m_0, m_1, \epsilon^*, t^*). \)

\( \mathcal{O}_T \) oracle query: It is simulated in the same way as that in Game 3.

Let \( Q_4 \) be the event that \( b' = b \) in Game 4. Similar to Equation (18), we have

\[
|\Pr[Q_4] - \Pr[Q_2]| \leq \Pr[E_2]. \tag{29}
\]

It is evident that all items in the challenge ciphertext and the trapdoor are independent of the message \( m \), and so we have

\[
\Pr[Q_4] = \frac{1}{2}. \tag{30}
\]

Next, we briefly describe some changes about the analysis of the probability of event \( E_2 \) in this modified game.

\[
\Pr[E_2] \leq \frac{q_D}{2^k} \cdot \text{Adv}_{\text{DDH}} + \frac{q_D}{2^k} + \text{Adv}_{\text{DDH}}. \tag{31}
\]

Lemma 5.5. Event \( E_2 \) happens in Game 4 with negligible probability if the BDH problem and the DDH problem are intractable.

Proof. The proof of \( E_2 \) is similar to Lemma 5.3. The difference is that we need to add simulated \( \mathcal{O}_T \) oracle for \( \mathcal{A} \) as follows.

\( \mathcal{O}_T \) oracle query: On input a public key \( pk_s = h \), which is generated by \( B \), it selects \( r_3 \in \mathbb{Z}_p^* \) and \( Z \in \mathbb{G}_1 \), and returns \((g_3^{r_3}, Z \cdot h^{r_1})\).

We denote the event \( D_3 \) that \( \mathcal{A} \) distinguishes the truly generated trapdoor and the simulated trapdoor because \( B \) cannot compute \( g_3^{r_3} \). Therefore, if both event \( D_4 \) and event \( D_3 \) do not happen, \((e(g_1, c_2)^y, \cdot) \) appears in some tuple on the \( H_1 \) table with probability at least \( \Pr[E_2] \). \( B \) obtains \((g, g^y) \) denoted by the solution of the BDH problem. This means that

\[
\Pr[\neg(D_4 \lor D_3)] = \left(1 - \frac{q_D}{2^k}\right) \Pr[\neg(E_2)],
\]

otherwise, \( B \) aborts with failure. Therefore, we have

\[
\Pr[\neg(D_4 \lor D_3)] = \Pr[\neg(D_4 \lor D_3)] \Pr[\neg(E_2)] \geq \frac{q_D}{2^k}\cdot \Pr[\neg(E_2)].
\]

Therefore,

\[
\text{Adv}_{\text{DDH}} \geq \frac{1}{q_D} \left( \Pr[E_2] - \frac{q_D}{2^k} - \text{Adv}_{\text{DDH}} \right). \tag{32}
\]

It is well known that \( \text{Adv}_{\text{DDH}} \) on \( \mathbb{G}_1 \) is negligible, and so if \( \Pr[E_2] \) is non-negligible, the probability of breaking the BDH assumption is non-negligible.

Owing to Equations (25)–(31), we claim that

\[
\text{Adv}_{\text{IND-CCA}}^\text{A-Type-H} = \text{Adv}_{\text{IND-CCA}}^\text{A-Type-H} + \frac{q_D}{2^k} + \frac{q_D}{2^k} + q_D\cdot \text{Adv}_{\text{DDH}} + \text{Adv}_{\text{DDH}}.
\]

This completes the proof of Theorem 5.3.

Theorem 5.4. The above PKE-DET scheme is Anon-TD secure according to Definition 3.6 under the DDH assumption.

Proof. We observe that the trapdoor is a ciphertext generated by the El Gamal encryption, and so it is easy to prove that the above PKE-DET scheme is Anon-TD secure under DDH assumption on \( \mathbb{G}_1 \). Let \( A \) be a PPT algorithm attacking the Anon-TD security of the above PKE-DET scheme. We construct a PPT algorithm \( B \) to break the DDH assumption on \( \mathbb{G}_1 \). Given \((g, g^y, g^z, g^z) \in \mathbb{G}_1^4 \) where \( z = xy \) or \( z \in R, \mathbb{Z}_p \), \( B \) does the following works.

1. Set the public parameters: \( g_3 = g \), and the server’s public key \( pk_s = g^z \).
2. Invoke \( A \) on input \( pk_s = g^z \) with the same distribution as that in PKE-DET and obtain two keys of data owners \( \mathcal{R}_0 \) and \( \mathcal{R}_1 \) from \( A \): \( \alpha_0 \) and \( \alpha_1 \).
This implies that a ciphertext construction achieves a significant security improvement. Given due to the desired functionality described in Section 3, our security improvement:

In [24], any entity can mount an Offline message recover attack. Tang [24] achieved a negligible function

$$\Pr[B(G_1, g, g^x, g^y, g^z) = 1] = 1 - \epsilon(\lambda).$$

Since the DDH problem on $G_1$ is hard, there must exist a negligible function

$$\text{negl}(\lambda) \geq 1 - \epsilon(\lambda).$$

This implies that $\epsilon(\lambda) \leq \frac{1}{2} + \text{negl}(\lambda)$, completing the proof.

Offline message recover attack. In [24], any entity can mount the attack since the equality test can be executed on any two ciphertexts without trapdoor information. Tang [24] achieved a security improvement:

1. The adversary without being delegated the equality test on ciphertexts cannot amount this type of attack.

2. The adversary delegated the equality test on ciphertexts can compute $g^m$ given the trapdoor. As a result, it needs to pre-compute $\{g^m | m' \in M\}$, and then the attack is a table lookup. Denoting by Exp the exponentiation operation, the computation complexity is $O(p) \cdot \text{Exp}$. Although offline message recovery attack is unavoidable due to the desired functionality described in Section 3, our construction achieves a significant security improvement. Given a ciphertext $c = \text{Encrypt}(m, pk_R)$ and a trapdoor $t = \text{Delegate}(sk_R, pk_S)$,

1. The adversary without being delegated the equality test on ciphertexts cannot amount this type of attack.

2. The adversary delegated the equality test on ciphertexts can compute $g^m = t_2/t_1^r$ and then $X = c_3/H_1(e(g_1^{t_2}, c_2))$ to obtain $m'$ given the trapdoor. As a result, the attack can be performed to check whether the equation $e(g_1^{*}, m') = e(g_1, m')$ is true for each $m' \in M$. Denote by Pairing the pairing operation, the computation complexity is $O(p) \cdot \text{Pairing}$.

Since the Pairing operation is much more expensive than the Exponent operation, we can claim that PKE-DET achieves a significant security improvement against offline message recovery attack.

6. COMPARISON

In Table 1, we compare PKE-DET proposed in Section 4 with PKEET [7], PKEET-UA [24] and PKE-DKS [22]. The second, third, fourth and fifth rows show the comparison for the data owner in algorithms $\text{KeyGen}_R$, $\text{Encrypt}$, $\text{Decrypt}$ and $\text{Delegate}$. The sixth and seventh rows indicate the computation comparison for the server in algorithms $\text{KeyGen}_S$ and $\text{Test}$. The eighth row lists the size of the ciphertext. The ninth and tenth rows show whether the schemes support keywords search or the equality test. The last five rows indicate that the comparison for the security, which includes the ciphertext security with or without the delegated equality test, whether achieving anonymous trapdoor security and whether resisting an offline attack with or without the delegated equality test (if not, the computation complexity of the adversary is shown).

In terms of the computation complexity, the major computational cost is due to the evaluation of bilinear pairings and scalar multiplications on elliptic curves, which are known to be costly and dominate the computational complexity. We choose to compare our PKE-DET scheme with that in [7, 22, 24] in terms of dominating computational operations. By means of an exponent array, simultaneous scalar multiplication of the form $g^Y$ is as expensive in computation as only 1.167 scalar multiplications [29]. According to the existing experimental results on ECC [30] and Pairing [31, 32], a bilinear pairing costs about five times than the elliptic curve scalar multiplication on a conventional desktop computer. Interested readers can refer to [30–33] for detailed information. The computational complexity of our protocol can be further improved by pre-computing $e(g_1; u_1)$ and including it in the public parameters. So the algorithm $\text{Encrypt}$ will need one less pairing evaluation. In terms of storage overheads, because PKEET [7], PKE-DKS [22] and PKE-DET are constructed on the elliptic curves for pairing, their ciphertexts are much shorter than that in PKEET-UA [24], which is constructed on the discrete logarithm group.

Compared with the first PKEET [7], PKE-DET has security improvement to prevent ciphertext comparison without the delegation. We make use of the first PKEET [7] and the well-known IBE [34] to obtain a PKE-DET construction. So it costs...
### TABLE 1. Comparison.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp</td>
<td>R</td>
<td>KeyGenR</td>
<td>KeyGenS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1S.(G)</td>
<td>2Exp.(G0)</td>
<td>2S.(G)</td>
</tr>
<tr>
<td>Encrypt</td>
<td>3S.(G)</td>
<td>5Exp.(G0)</td>
<td>2S.(G)</td>
<td>5S.(G)</td>
</tr>
<tr>
<td>Decrypt</td>
<td>3S.(G)</td>
<td>2Exp.(G0)</td>
<td>1S.(G)</td>
<td></td>
</tr>
<tr>
<td>Delegate</td>
<td>—</td>
<td>0</td>
<td>4S.(G)+1SM.(G) ≈ 5.167S.(G)</td>
<td>1S.(G)+1SM.(G) ≈ 2.167S.(G)</td>
</tr>
<tr>
<td>S</td>
<td>KeyGenS</td>
<td>—</td>
<td>1S.(G)</td>
<td>1S.(G)</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td>2P</td>
<td>4Exp.(G0)</td>
<td>3P</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2P</td>
<td>2S.(G)+4P</td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td>3G + Zp ≈ 4k</td>
<td>3G0 + 2Zp0 +</td>
<td>H3</td>
</tr>
<tr>
<td>Functionality</td>
<td>Keyword search</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>Equality test</td>
<td>√</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Security</td>
<td>Ciphertext with Del</td>
<td>—</td>
<td>OW-CCA</td>
<td>OW-CCA</td>
</tr>
<tr>
<td></td>
<td>Ciphertext w/o Del</td>
<td>OW-CCA</td>
<td>IND-CCA</td>
<td>IND-CCA</td>
</tr>
<tr>
<td></td>
<td>Anon-TD</td>
<td>—</td>
<td>×</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Off-Att with Del</td>
<td>—</td>
<td>O(p) · Exp</td>
<td>O(p) · P</td>
</tr>
<tr>
<td></td>
<td>Off-Att w/o Del</td>
<td>O(p) · P</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Comp, computation; R, the data owner; S, the cloud server; P, bilinear pairing (e.g. e(g1, g2)); S, scalar multiplication (e.g. g^s); SM, simultaneous scalar multiplication (e.g. g^sY); Exp, an exponentiation operation in a discrete logarithm group; (G, p), a group G of prime order p for pairing; (G0, p0), a discrete logarithm group G0 of prime order p0; Del, delegation; w/o, without; Off-Att, offline attack.

Security parameters: k = |G| ≈ |Zp| = 224, |G0| = 2048, |Zp0| = 224, |H3| = 224 (recommended by NIST [28]).
more computation overheads. The goal of PKE-DET is similar to that of PKEET-UA [24]. The difference is that PKE-DET can achieve anonymous trapdoor security and has more advantage on resisting offline message recovery attack. However, another problem arises that the computation overheads of the legal test on ciphertexts also increases. Since the legal test is run by the cloud server, which has more powerful computing capability, PKE-DET is suitable and applicable for the scenario with higher security requirement. Compared with PKE-DKS [22], PKE-DET supports more functionality than PKE-DKS and has higher ciphertext security.

In conclusion, we do not need to sacrifice much efficiency in algorithms of KeyGenS, KeyGenR and Encrypt than that in [7, 22, 24]. Owing to the powerful computation capability of the cloud server, the loss in computation efficiency in the Test algorithm is tolerable as we now have security improvement.

7. CONCLUSION

In this paper, we propose a PKE-DET in a multi-user setting and prove in a random oracle model that our scheme achieves: (1) OW-CCA secure against a Type-I adversary under the CDH and co-DH assumptions; (2) IND-CCA secure against a Type-II adversary under the CDH and BDH assumptions; (3) IND-CCA secure against a Type-III adversary under the CDH, BDH and DDH assumptions; (4) Anon-TD secure under the DDH assumption. We also compare it with other schemes [7, 22, 24] with regard to the computation overheads of receiver and server, the storage overheads, the supported functionality and the security.

FUNDING

This work is supported by the National Natural Science Foundation of China (No. 61370224, 61103232, 61272436 and 61272404), Key Program of Natural Science Foundation of Hubei Province (No. 2013CFA046), the Open Fund Program for State Key Laboratory of Information Security, the Guangdong Natural Science Foundation (No. S2013010011859 and 1035180601000000) and the Research Fund for the Doctoral Program of Higher Education of China (No. 20114404120027).

REFERENCES


