Public Key Encryption Supporting Equality Test and Flexible Authorization without Bilinear Pairings

Xi-Jun Lin *, Haipeng Qu †and Xiaoshuai Zhang ‡

March 11, 2016

Abstract: In recent years, public key encryption with equality test (PKEET) has become a hot research topic in the cryptography community due to the advancement of cloud computing. Recently, Ma et al. proposed a new related primitive, called public key encryption with equality test supporting flexible authorization (PKEET-FA), and constructed a concrete scheme. In their proposal, four types of authorization were presented to support different authorization policies. However, their proposal is based on bilinear pairings which are time costing operations compared with modular exponentiations. In this paper, we present a new PKEET-FA scheme without bilinear pairings. Compared with the existing work, our proposal is more efficient.

Key words: searchable encryption; public key encryption with equality test; flexible authorization

1 Introduction

In 2010, the notion of public key encryption with equality test (PKEET) was first proposed by Yang et al. [21]. In their proposal, anyone has the ability to check whether or not two ciphertexts contain the same message. However, the adversary is able to check the equality of ciphertexts without any authorization. To protect data owners' privacy, an authorization mechanism has to be used. For this reason, a fine-grained authorization policy enforcement mechanism was integrated into PKEET by Tang [20], and then an enhanced notion, called PKEET with fine-grained authorization (FG-PKEET), was introduced. Tang [18] also presented a new scheme, which was extended from FG-PKEET, by using two collaborating proxies to perform the equality tests. Moreover, an all-or-nothing PKEET (AoN-PKEET) was presented by Tang [19], where a coarse-grained authorization mechanism was proposed to specify who is able to perform the equality tests.

On the other hand, all above schemes are one-way against chosen-ciphertext attack (OW-CCA) in the random oracle model. However, for some special scenarios, such as database applications, OW-CCA security may not be strong enough. Motivated by this, the security models of PKEET were revisited by Lu et al. [12], who proposed several new and stronger

*X.J.Lin is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China. email: linxj77@163.com

†H.Qu is with the Department of Computer Science and Technology, Ocean University of China. Qingdao 266100, P.R.China.

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

security definitions. A new notion, called public key encryption with delegated equality test (PKE-DET), was introduced by Ma et al. [15], where the delegated party is only allowed to deal with the work in a multi-user setting.

There are also several related notions. Bellare et al. [2] presented the notion of deterministic encryption (DE). Bellare et al. [3] improved DE by using general assumptions, which was followed by an efficient construction without random oracles by Boldyreva et al. [4] and a framework for the security of DE by Brakerski et al. [6]. The equality test could be trivially achieved by DE on ciphertexts encrypted under the same public key. However, DE is a deterministic algorithm, while PKEET is a probabilistic algorithm. Moreover, the equality test is not performed on ciphertexts with respect to same public keys in PKEET, but different public keys. Public key encryption with keyword search (PEKS) is another related notion presented by Boneh et al. [5], which supports keyword searching over ciphertexts without retrieving messages by using corresponding trapdoors. Since then, plenty of proposals with additional functionalities were presented [1, 7–10, 16, 17, 22]. In fact, PKEET, which trivially supports the traditional functionality of PEKS, is an extension of PEKS.

Recently, Ma et al. [14] proposed a new primitive, called public key encryption with equality test supporting flexible authorization (PKEET-FA), to strengthen the privacy protection in four scenarios with different authorization granularity levels. An efficient construction of PKEET-FA was also presented in [14], where the following four different authorization policies are introduced.

- Type-1. User level authorization: Alice's all ciphertexts could be compared with all ciphertexts of any other receiver.
- Type-2. Ciphertext level authorization: An Alice's specific ciphertext could be compared with a specific ciphertext of any other receiver.
- Type-3. User-specific ciphertext level authorization: An Alice's specific ciphertext could be only compared with a specific ciphertext of a specific receiver for example, Bob, but could not be compared with any ciphertext of any receiver other than Bob.
- Type-4. Ciphertext-to-user (or User-to-ciphertext) level authorization: An Alice's specific ciphertext could be compared with all ciphertexts of any other receiver, vice versa.

In fact, Type-4 authorization is a combination of Type-1 authorization and Type-2 authorization, which is for comparing a single ciphertext of Alice with all ciphertexts of any other receiver.

1.1 Our Contribution

The main technique for equality test in most existing work is based on bilinear pairings. However, compared with modular exponentiations, bilinear pairings are still time costing operations although some efforts have been made to improve the efficiency. Note that a bilinear pairing costs about five times than a modular exponentiation in a conventional desktop computer [11, 13, 23]. To the best of our knowledge, the only PKEET without bilinear pairings was proposed in [19]. However, compared with PKEET-FA, the scheme

has less flexibility. Hence, we believe that a PKEET-FA without bilinear pairings would enjoy both efficiency and flexibility, which is more suitable for practice.

Motivated by this, we present in this paper a new PKEET-FA without bilinear pairings. The main approach in our proposal for improving the efficiency of the equality tests is based on the property of Shamir's secret shairing. Compared with the state-of-the-art scheme [14], our proposal is more efficient, especially in terms of the equality test. And compared with the other existing work [18–21], our proposal enjoys both efficiency and flexibility. More details can be referred to Section 6 for efficiency comparisons.

1.2 Organization

The rest of the paper is organized as follows: In Section 2, we recall some definitions, such as Decision Diffie-Hellman problem and Shamir's secret sharing scheme. The system model, the definition and the security models of PKEET-FA are recalled in Section 3. The proposed new scheme is shown in Section 4. Then, the security proofs of our proposal are given in Section 5. The efficiency comparisons are shown in Section 6, which is followed by the last section to conclude our work.

2 Preliminaries

2.1 Decision Diffie-Hellman (DDH) Problem

Decision Diffie-Hellman (DDH) Problem: Let $\mathbb{G}$ be a group of prime order q. The DDH problem is ϵ-hard in $\mathbb{G}$, if given $(g, g^a, g^b, h) \in \mathbb{G}^4$ as input for random generator $g \in_R \mathbb{G}$, $a, b \in_R \mathbb{Z}_q$ and $h \in_R \mathbb{G}$, any probabilistic polynomial-time algorithm $\mathcal{A}$ decides whether or not $h = g^{ab}$ holds with advantage

$$Adv^{DDH}_{\mathcal{A},\mathbb{G}} = |Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - Pr[\mathcal{A}(g, g^a, g^b, h) = 1]| \leq \epsilon.$$

We say that DDH assumption holds if for any probabilistic polynomial-time algorithm $\mathcal{A}$, its advantage $Adv^{DDH}_{\mathcal{A},\mathbb{G}}$ is negligible.

2.2 Shamir's Secret Sharing Scheme

In a secret sharing scheme, a secret y is divided into n shares and shared among n shareholders in such a way that, with any t or more than t shares, it is able to reconstruct this secret; but, with fewer than t shares, it cannot reconstruct the secret.

Shamir's secret sharing scheme is based on Lagrange interpolating polynomial, and there are n shareholders $\mathcal{U} = \{U_1, \cdots, U_n\}$. The scheme consists of the following algorithms:

- SSharing.Setup(λ): This algorithm takes as input the security parameter λ, and outputs the public parameter $pp = q$, where q is a large random prime.
- SSharing.Generation(pp, y): This algorithm takes as inputs the public parameter $pp = q$ and the secret $y \in \mathbb{Z}_q$, and does the following:
  1. Pick a polynomial $f(x)$ of degree $t-1$ randomly: $f(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1} \pmod p$, where the secret $y = a_0 = f(0)$ and all coefficients $a_0, a_1, \cdots, a_{t-1}$ are in $\mathbb{Z}_q$.

2. Compute all shares: $y_i = f(x_i) \pmod q$ for $i = 1, \cdots, n$, where $x_i \in \mathbb{Z}_q$ are picked randomly.
3. Output a list of n points $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$.

Each share y_i is distributed to corresponding shareholder U_i privately. Note that x_i need not be kept secretly.

- SSharing.Reconstruction$(pp, (x_{i_1}, y_{i_1}), \cdots, (x_{i_t}, y_{i_t}))$: This algorithm takes the public parameter $pp = q$ and any t points $\{(x_{i_1}, y_{i_1}), \cdots, (x_{i_t}, y_{i_t})\}$ as inputs, it reconstructs and outputs the secret y as

$$y = f(0) = \sum_{i \in A} \Delta_i y_i \pmod q,$$

where $\Delta_i = \prod_{j \in A/\{i\}} \frac{x_j}{x_j - x_i} \pmod q$ are the Lagrange interpolation coefficients and $A = \{i_1, \cdots, i_t\} \subseteq \{1, \cdots, n\}$.

2.3 Notation

Let $[x]_a^b$ denote the substring taken from the a-th bit to the b-th bit of x, where x is a bit-string.

3 System Model and Definition

In this section, we recall the system model and definition for PKEET-FA presented in [14].

3.1 System Model

In a PKEET-FA system, there are four entities: a cloud server, two receivers (for example, Alice and Bob) and a trusted third party (denoted by TTP). TTP's task is to setup the system and generate the system public parameters for receivers and the cloud server. Anyone can encrypt message M_A (resp. M_B) to Alice (resp. Bob) by using the corresponding public key. The ciphertexts C_A (resp. C_B) is outsourced to the could server. Alice and Bob have the ability to authorize the cloud server to test the equality on M_A and M_B without decrypting C_A and C_B.

3.2 Definition

Definition 1 *A PKEET-FA scheme consists of the following algorithms:*

- *Setup(λ)*: This algorithm takes as input the security parameter λ and outputs the public parameter *pp*.
- *KeyGen(pp)*: This algorithm takes as input the public parameter *pp*, and outputs a public/private key pair (pk, sk).

- *Encrypt*(M, pk): This algorithm takes as input a message M and the receiver's public key pk, and outputs a ciphertext C.
- *Decrypt*(C, sk): This algorithm takes as input a ciphertext C and the receiver's private key sk, and outputs a message M (or an error symbol $\perp$).

Suppose that the receiver U_i (resp. U_j) has a public/private key pair (pk, sk) (resp. (pk', sk')), whose ciphertext is C (resp. C'). To realize Type-π ($\pi = 1, 2, 3, 4$) authorization for U_i and U_j, algorithm Aut_π ($\pi = 1, 2, 3, 4$) is defined to generate trapdoor for C which is required to be compared with C', and algorithm $Test_\pi$ ($\pi = 1, 2, 3, 4$) is defined to determine whether or not C and C' contain the same message.

- *Type*-1 *Authorization*:
  - $Aut_1(sk)$: This algorithm takes as input U_i's private key sk, and outputs a trapdoor td_1 for U_i.
  - $Test_1(C, td_1, C', td'_1)$: This algorithm takes as inputs ciphertext C, the trapdoor td_1, ciphertext C' and the trapdoor td'_1, and outputs 1 if C and C' contain the same message and 0 otherwise.
- *Type*-2 *Authorization*:
  - $Aut_2(sk, C)$: This algorithm takes as inputs U_i's private key sk and ciphertext C, and outputs a trapdoor td_2 for (U_i, C).
  - $Test_2(C, td_2, C', td'_2)$: This algorithm takes as inputs ciphertext C, the trapdoor td_2, ciphertext C' and the trapdoor td'_2, and outputs 1 if C and C' contain the same message and 0 otherwise.
- *Type*-3 *Authorization*:
  - $Aut_3(sk, C, C')$: This algorithm takes as inputs U_i's private key sk, ciphertext C and ciphertext C', and outputs a trapdoor td_3 for (U_i, C, C').
  - $Test_3(C, td_3, C', td'_3)$: This algorithm takes as inputs ciphertext C, the trapdoor td_3, ciphertext C' and the trapdoor td'_3, and outputs 1 if C and C' contain the same message and 0 otherwise.
- *Type*-4 *Authorization*:
  - $Aut_4(sk, C)$: This algorithm takes as inputs U_i's private key sk and ciphertext C, and outputs a trapdoor $td_4 = Aut_2(sk, C)$ for (U_i, C).
  - $Aut'_4(sk')$: This algorithm takes as input U_j's private key sk', and outputs a trapdoor $td'_4 = Aut_1(sk')$ for U_j.
  - $Test_4(C, td_4, C', td'_4)$: This algorithm takes as inputs ciphertext C, the trapdoor td_4, ciphertext C' and the trapdoor td'_4, and outputs 1 if C and C' contain the same message and 0 otherwise.

Definition 2 (Correctness) *A PKEET-FA scheme is correct if for any* $pp \leftarrow Setup(\lambda)$, $(pk, sk) \leftarrow KeyGen(pp)$ *and* $(pk', sk') \leftarrow KeyGen(pp)$, *the following conditions are satisfied.*

1. For any M, $Decrypt(Encrypt(M, pk), sk) = M$ always holds.
2. For any ciphertexts C and C', if $Decrypt(C, sk) = Decrypt(C', sk') \neq \perp$:
   - Type-1 Authorization: Given $Aut_1(sk) = td_1$ and $Aut_1(sk') = td'_1$, it holds that
   $$Test_1(C, td_1, C', td'_1) = 1.$$
   - Type-2 Authorization: Given $Aut_2(sk, C) = td_2$ and $Aut_2(sk', C') = td'_2$, it holds that
   $$Test_2(C, td_2, C', td'_2) = 1.$$
   - Type-3 Authorization: Given $Aut_3(sk, C, C') = td_3$ and $Aut_3(sk', C', C) = td'_3$, it holds that
   $$Test_3(C, td_3, C', td'_3) = 1.$$
   - Type-4 Authorization: Given $Aut_4(sk, C) = td_4$ and $Aut'_4(sk') = td'_4$, it holds that
   $$Test_4(C, td_4, C', td'_4) = 1.$$
3. For any ciphertexts C and C', if $Decrypt(C, sk) \neq Decrypt(C', sk')$:
   - Type-1 Authorization: Given $Aut_1(sk) = td_1$ and $Aut_1(sk') = td'_1$, it holds that $Pr[Test_1(C, td_1, C', td'_1) = 1]$ is negligible.
   - Type-2 Authorization: Given $Aut_2(sk, C) = td_2$ and $Aut_2(sk', C') = td'_2$, it holds that $Pr[Test_2(C, td_2, C', td'_2) = 1]$ is negligible.
   - Type-3 Authorization: Given $Aut_3(sk, C, C') = td_3$ and $Aut_3(sk', C', C) = td'_3$, it holds that $Pr[Test_3(C, td_3, C', td'_3) = 1]$ is negligible.
   - Type-4 Authorization: Given $Aut_4(sk, C) = td_4$ and $Aut'_4(sk') = td'_4$, it holds that $Pr[Test_4(C, td_4, C', td'_4) = 1]$ is negligible.

3.3 Security Models

Here, we recall the security models for PKEET-FA defined in [14]. Note that only Type-π ($\pi = 1, 2, 3$) authorization queries are provided to the adversary for simplicity (Type-4 authorization queries are omitted here), since the trapdoor algorithm for Type-4 authorization in fact is a combination of those for Type-1 and Type-2 authorizations.

Formally, there are two types of adversaries defined for the security of PKEET-FA:

- Type-I adversary: For Type-π ($\pi = 1, 2, 3$) authorization, the adversary with the Type-π trapdoor cannot retrieve the message from the challenge ciphertext.
- Type-II adversary: For Type-π ($\pi = 1, 2, 3$) authorization, the adversary without the Type-π trapdoor cannot decide the challenge ciphertext is the encryption of which message.

The formal definition of OW-CCA security for Type-π ($\pi = 1, 2, 3$) authorization against Type-I adversary is recalled below.

Game 1 *Let $\mathcal{A}_1$ be a Type-I adversary. The target receiver has index t ($1 \leq t \leq n$). The game between $\mathcal{A}_1$ and the challenger $\mathcal{C}$ is as follows:*

1. *Setup*: $\mathcal{C}$ takes a security parameter λ and runs the algorithm *Setup* to generate the public parameter *pp* which is given to $\mathcal{A}_1$. Then $\mathcal{C}$ runs algorithm *KeyGen* to generate n public/private key pairs (pk_i, sk_i) $(1 \leq i \leq n)$ and gives all pk_i to $\mathcal{A}_1$.

2. *Phase* 1: $\mathcal{A}_1$ is allowed to issue the following queries for polynomially many times. The constraint is that (t) does not appear in the key retrieve queries.

   - Key retrieve queries (i): $\mathcal{C}$ sends sk_i to $\mathcal{A}_1$.
   - Decryption queries (i, C): $\mathcal{C}$ runs $Decrypt(C, sk_i)$ to decrypt C with sk_i, and sends the output to $\mathcal{A}_1$.
   - Authorization queries: For Type-π $(\pi = 1, 2, 3)$ authorization,
     - on input (i), $\mathcal{C}$ returns td_1;
     - on input (i, C), $\mathcal{C}$ returns td_2;
     - on input (i, C, C'), $\mathcal{C}$ returns td_3.

3. *Challenge*: $\mathcal{C}$ picks a message M_t randomly, computes the challenge ciphertext $C_t^* = Encrypt(M_t, pk_t)$ and sends C_t^* to $\mathcal{A}_1$.

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

   (a) (t) does not appear in the key retrieve queries;

   (b) (t, C_t^*) does not appear in the decryption queries.

5. *Guess*: $\mathcal{A}_1$ outputs M_t', and wins the game if $M_t = M_t'$.

The advantage of $\mathcal{A}_1$ in the game above is defined as

$$Adv_{PKEET-FA,\mathcal{A}_1}^{OW-CCA,Type-\pi}(\lambda) = Pr[M_t = M_t'] \quad (\pi = 1, 2, 3).$$

Definition 3 *A PKEET-FA scheme is OW-CCA secure for Type-π ($\pi = 1, 2, 3$) authorization if for any probabilistic polynomial-time adversary $\mathcal{A}_1$, its advantage $Adv_{PKEET-FA,\mathcal{A}_1}^{OW-CCA,Type-\pi}(\lambda)$ is negligible on λ.*

The formal definition of IND-CCA security for Type-π $(\pi = 1, 2, 3)$ authorization against Type-II adversary is recalled below.

Game 2 *Let $\mathcal{A}_2$ be a Type-II adversary. The target receiver has index t $(1 \leq t \leq n)$. The game between $\mathcal{A}_2$ and the challenger $\mathcal{C}$ is as follows:*

1. *Setup*: $\mathcal{C}$ takes a security parameter λ and runs the algorithm *Setup* to generate the public parameter *pp* which is given to $\mathcal{A}_2$. Then $\mathcal{C}$ runs algorithm *KeyGen* to generate n public/private key pairs (pk_i, sk_i) $(1 \leq i \leq n)$ and gives all pk_i to $\mathcal{A}_2$.

2. *Phase* 1: $\mathcal{A}_2$ issues queries as in *Game* 1.

3. *Challenge*: $\mathcal{A}_2$ submits two equal-length messages M_0, M_1. $\mathcal{C}$ picks a random bit $\rho \in \{0, 1\}$, computes the challenge ciphertext $C_t^* = Encrypt(M_\rho, pk_t)$ and sends C_t^* to $\mathcal{A}_2$.

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

- (t) does not appear in the key retrieve queries;
- (t, C_t^*) does not appear in the decryption queries;
- For Type-π ($\pi = 1, 2, 3$) authorization queries,
  - $\pi = 1$: (t) does not appear in the authorization queries;
  - $\pi = 2$: (t, C_t^*) does not appear in the authorization queries;
  - $\pi = 3$: $(t, C_t^*, \cdot)$ does not appear in the authorization queries.

5. *Guess*: $\mathcal{A}_2$ outputs a guess $\rho' \in \{0, 1\}$, and wins the game if $\rho = \rho'$.

The advantage of $\mathcal{A}_2$ in the game above is defined as

$$Adv_{PKEET-FA,\mathcal{A}_2}^{IND-CCA,Type-\pi}(\lambda) = |Pr[\rho = \rho'] - \frac{1}{2}| \quad (\pi = 1, 2, 3).$$

Definition 4 *A PKEET-FA scheme is IND-CCA secure for Type-π ($\pi = 1, 2, 3$) authorization if for any probabilistic polynomial-time adversary $\mathcal{A}_2$, its advantage $Adv_{PKEET-FA,\mathcal{A}_2}^{IND-CCA,Type-\pi}(\lambda)$ is negligible on λ.*

4 Our Proposed PKEET-FA Scheme

In this section, we propose our new and efficient PKEET-FA scheme.

4.1 The Proposed Scheme

- *Setup(λ)*: This algorithm takes as input security parameter λ, and outputs public parameters *pp* as follows.

  1. Generate a group $\mathbb{G}$ of prime order q.
  2. Pick a random generator $g \in \mathbb{G}$.
  3. Select cryptographic secure hash functions: $H : \mathbb{G} \to \{0,1\}^{\lambda+l'}$, $H_1 : \mathbb{G}^3 \times \{0,1\}^{\lambda+l'} \to \{0,1\}^{2l+2l'}$, and $H_2, H_3, H_4, H_5, H_6, H_7 : \{0,1\}^{\lambda} \to \mathbb{Z}_q$, where $\{0,1\}^l \subset \mathbb{Z}_q$ and l' is representation length of elements in $\mathbb{Z}_q$.

- *KeyGen(pp)*: This algorithm takes as input public parameter *pp*, then picks $\alpha, \beta \in \mathbb{Z}_q$ randomly and outputs the receiver's key pair:

$$(pk, sk) = ((X = g^{\alpha}, Y = g^{\beta}), (\alpha, \beta)).$$

- *Encrypt(M, pk)*: This algorithm takes as inputs a message $M \in \{0,1\}^{\lambda}$ and the receiver's public key *pk*, then generates a ciphertext $C = (C_1, C_2, C_3, C_4)$ as follows:

  1. Compute three points $P_1 = (H_2(M), H_3(M))$, $P_2 = (H_4(M), H_5(M))$ and $P_3 = (H_6(M), H_7(M))$.
  2. Construct an interpolating polynomial $f(x)$ with degree 2 to pass through three points: P_1, P_2 and P_3.
  3. Compute two additional random points (x_1, y_1) and (x_2, y_2) on $f(x)$, where $x_1, x_2 \in \{0,1\}^l$.

4. Pick two random numbers $r_1, r_2 \in \mathbb{Z}_q$ and then compute

$$\begin{aligned} C_1 &= g^{r_1} \\ C_2 &= g^{r_2} \\ C_3 &= (M\|r_2) \oplus H(Y^{r_1}) \\ C_4 &= (x_1\|x_2\|y_1\|y_2) \oplus H_1(X^{r_2}, C_1, C_2, C_3) \end{aligned}$$

- *Decrypt*(C, sk): This algorithm takes as inputs a ciphertext $C = (C_1, C_2, C_3, C_4)$ and receiver's private key sk, then performs the following steps:

  1. Recover $M\|r_2$ by computing $C_3 \oplus H(C_1^{\beta})$.
  2. Recover $x_1\|x_2\|y_1\|y_2$ by computing $C_4 \oplus H_1(C_2^{\alpha}, C_1, C_2, C_3)$.
  3. Compute P_1, P_2 and P_3 as done in algorithm *Encrypt*.
  4. Construct $f(x)$ with P_1, P_2 and P_3 as done in algorithm *Encrypt*.

  If $C_2 = g^{r_2}$, $f(x_1) = y_1$ and $f(x_2) = y_2$ all hold, this algorithm outputs M; otherwise, it outputs $\perp$.

Let Alice (whose public/private key pair is (pk, sk)) and Bob (whose public/private key pair is (pk', sk')) be two receivers in the system. Let $C = (C_1, C_2, C_3, C_4)$ (resp. $C' = (C_1', C_2', C_3', C_4')$) be a ciphertext of Alice (resp. Bob). Correspondingly, let r_1, r_2 (resp. r_1', r_2') be the randomness used for generating C (resp. C').

- *Type*-1 *Authorization*:
  - $Aut_1(sk)$: This algorithm outputs a trapdoor $td_1 = \alpha$.
  - $Test_1(C, td_1, C', td_1')$: This algorithm computes

$$\begin{aligned} C_4 \oplus H_1(C_2^{td_1}, C_1, C_2, C_3) &= x_1\|x_2\|y_1\|y_2, \\ C_4' \oplus H_1(C_2^{'td_1'}, C_1', C_2', C_3') &= x_1'\|x_2'\|y_1'\|y_2'. \end{aligned}$$

    Then, it computes

$$\begin{aligned} &\varphi \leftarrow SSharing.Reconstruction(q, (x_1, y_1), (x_2, y_2), (x_1', y_1')), \\ &\varphi' \leftarrow SSharing.Reconstruction(q, (x_1', y_1'), (x_2', y_2'), (x_1, y_1)). \end{aligned}$$

    Finally, it checks whether or not $\varphi = \varphi'$ holds. If it is the case, it returns 1, and 0 otherwise.

- *Type*-2 *Authorization*:
  - $Aut_2(sk, C)$: This algorithm outputs a trapdoor $td_2 = H_1(C_2^{\alpha}, C_1, C_2, C_3)$.
  - $Test_2(C, td_2, C', td_2')$: This algorithm computes

$$\begin{aligned} C_4 \oplus td_2 &= x_1\|x_2\|y_1\|y_2, \\ C_4' \oplus td_2' &= x_1'\|x_2'\|y_1'\|y_2'. \end{aligned}$$

    Then, it computes

$$\begin{aligned} &\varphi \leftarrow SSharing.Reconstruction(q, (x_1, y_1), (x_2, y_2), (x_1', y_1')), \\ &\varphi' \leftarrow SSharing.Reconstruction(q, (x_1', y_1'), (x_2', y_2'), (x_1, y_1)). \end{aligned}$$

Finally, it checks whether or not $\varphi = \varphi'$ holds. If it is the case, it returns 1, and 0 otherwise.

- *Type*-3 *Authorization*:
  - $Aut_3(sk, C, C')$: This algorithm first recovers r_2, y_1, y_2 with sk, and then outputs a trapdoor
$$td_3 = (z, V_1, V_2) = ([H_1(C_2^{\alpha}, C_1, C_2, C_3)]_0^{2l-1}, W^{y_1}, W^{y_2}),$$
where $W = C_2^{'r_2}$.
  - $Test_3(C, td_3, C', td_3')$: This algorithm computes
$$[C_4]_0^{2l-1} \oplus z = x_1 \| x_2,$$
$$[C_4']_0^{2l-1} \oplus z' = x_1' \| x_2'.$$
Then, it computes the Lagrange interpolation coefficients $\Delta_i = \prod_{j \in A/\{i\}} \frac{x_j}{x_j - x_i} \pmod q$ (resp. $\Delta_i' = \prod_{j \in A/\{i\}} \frac{x_j'}{x_j' - x_i'} \pmod q$) with respect to $x_1, x_2, x_3 = x_1'$ (resp. $x_1', x_2', x_3' = x_1$), where $A = \{1, 2, 3\}$, $i \in A$.
Finally, it checks whether or not $\prod_{i=1}^{3} V_i^{\Delta_i} = \prod_{j=1}^{3} V_j^{'\Delta_j'}$ holds, where $V_3 = V_1'$ and $V_3' = V_1$. If it is the case, it returns 1, and 0 otherwise.

- *Type*-4 *Authorization*:
  - $Aut_4(sk, C)$: This algorithm outputs a trapdoor $td_4 = Aut_2(sk, C) = H_1(C_2^{\alpha}, C_1, C_2, C_3)$.
  - $Aut_4'(sk')$: This algorithm outputs a trapdoor $td_4' = Aut_1(sk') = \alpha'$.
  - $Test_4(C, td_4, C', td_4')$: This algorithm computes
$$\begin{aligned} C_4 \oplus td_4 &= x_1 \| x_2 \| y_1 \| y_2, \\ C_4' \oplus H_1(C_2^{'td_4'}, C_1', C_2', C_3') &= x_1' \| x_2' \| y_1' \| y_2'. \end{aligned}$$
Then, it computes
$$\varphi \leftarrow SSharing.Reconstruction(q, (x_1, y_1), (x_2, y_2), (x_1', y_1')),$$
$$\varphi' \leftarrow SSharing.Reconstruction(q, (x_1', y_1'), (x_2', y_2'), (x_1, y_1)).$$
Finally, it checks whether or not $\varphi = \varphi'$ holds. If it is the case, it returns 1, and 0 otherwise.

4.2 Correctness

Theorem 1 *The above PKEET-FA scheme is correct according to Definition 2.*

Proof : we show that the three conditions are all satisfied.

1. The first condition holds naturally.

2. As for the second condition, we have the following fact:

   For any message M (resp. M'), the interpolating polynomial $f(x)$ (resp. $f'(x)$) with degree 2 is constructed by passing through three points $P_1 = (H_2(M), H_3(M))$, $P_2 = (H_4(M), H_5(M))$ and $P_3 = (H_6(M),\ H_7(M))$ (resp. $P'_1 = (H_2(M'), H_3(M'))$, $P'_2 = (H_4(M'), H_5(M'))$ and $P'_3 = (H_6(M'),\ H_7(M'))$).

   Clearly, $f(x) = f'(x)$ if $M = M'$. Hence, for any points (x_1, y_1) and (x_2, y_2) on $f(x)$, and (x'_1, y'_1) and (x'_2, y'_2) on $f'(x)$, we can claim that $(x_1, y_1), (x_2, y_2), (x'_1, y'_1), (x'_2, y'_2)$ are all valid points on both $f(x)$ and $f'(x)$ if $M = M'$.

   From above and the property of Shamir's secret sharing, if $M = M'$, then we have the following facts: Given $(x_1, y_1), (x_2, y_2)$ and (x'_1, y'_1), the value $f(0)$ could be reconstructed, and given $(x'_1, y'_1), (x'_2, y'_2)$ and (x_1, y_1), the value $f'(0)$ could be reconstructed as well. Moreover, $f(0) = f'(0)$ in this case.

   Hence, for any two public/private key pairs (pk, sk) and (pk', sk'), and ciphertexts $C = \text{Encrypt}(M, pk)$ and $C' = \text{Encrypt}(M', pk')$, the followings hold.

   - *Type*-1 *Authorization*: Given $td_1 = \alpha$ and $td'_1 = \alpha'$, we have

     $$\begin{aligned} C_4 \oplus H_1(C_2^{td_1}, C_1, C_2, C_3) &= x_1\|x_2\|y_1\|y_2, \\ C'_4 \oplus H_1(C_2'^{td'_1}, C'_1, C'_2, C'_3) &= x'_1\|x'_2\|y'_1\|y'_2. \end{aligned}$$

     From above, we have that if $M = M'$, then $\varphi = f(0)$ and $\varphi' = f'(0)$. Hence, we have that $\varphi = \varphi'$ holds if $M = M'$.

   - *Type*-2 *Authorization*: Given $td_2 = H_1(C_2^{\alpha}, C_1, C_2, C_3)$ and $td'_2 = H_1(C_2'^{\alpha'}, C'_1, C'_2, C'_3)$, we have

     $$\begin{aligned} C_4 \oplus td_2 &= x_1\|x_2\|y_1\|y_2, \\ C'_4 \oplus td'_2 &= x'_1\|x'_2\|y'_1\|y'_2. \end{aligned}$$

     From above, we have that if $M = M'$, then $\varphi = f(0)$ and $\varphi' = f'(0)$. Hence, we have that $\varphi = \varphi'$ holds if $M = M'$.

   - *Type*-3 *Authorization*: Note that

     $$td_3 = (z, V_1, V_2) = ([H_1(C_2^{\alpha}, C_1, C_2, C_3)]_0^{2l-1}, W^{y_1}, W^{y_2})$$

     and

     $$td'_3 = (z', V'_1, V'_2) = ([H_1(C_2'^{\alpha'}, C'_1, C'_2, C'_3)]_0^{2l-1}, W'^{y'_1}, W'^{y'_2}),$$

     where $W = C_2'^{r_2}$ and $W' = C_2^{r'_2}$.
     Clearly, $W = W' = g^{r'_2 r_2}$. Therefore,

     $$td'_3 = (z', V'_1, V'_2) = ([H_1(C_2'^{\alpha'}, C'_1, C'_2, C'_3)]_0^{2l-1}, W^{y'_1}, W^{y'_2}).$$

     Given td_3 and td'_3, we have

     $$[C_4]_0^{2l-1} \oplus z = x_1\|x_2,$$

     $$[C'_4]_0^{2l-1} \oplus z' = x'_1\|x'_2.$$

Moreover, we have that $(x_3, V_3) = (x_1', W^{'y_1'}) = (x_1', W^{y_1'})$ and $(x_3', V_3') = (x_1, W^{y_1})$. Then,

$$\begin{aligned}\prod_{i=1}^{3} V_i^{\Delta_i} &= V_1^{\Delta_1} V_2^{\Delta_2} V_3^{\Delta_3} \\ &= W^{\Delta_1 y_1} W^{\Delta_2 y_2} W^{\Delta_3 y_1'} \\ &= W^{\Delta_1 y_1 + \Delta_2 y_2 + \Delta_3 y_1'}\end{aligned}$$

$$\begin{aligned}\prod_{j=1}^{3} V_j^{'\Delta_j'} &= V_1^{'\Delta_1'} V_2^{'\Delta_2'} V_3^{'\Delta_3'} \\ &= W^{\Delta_1' y_1'} W^{\Delta_2' y_2'} W^{\Delta_3' y_1} \\ &= W^{\Delta_1' y_1' + \Delta_2' y_2' + \Delta_3' y_1}\end{aligned}$$

From above and the property of Shamir's secret sharing, we can conclude that if $M = M'$ (note that $f(0) = f'(0)$ in this case), then we have

$$f(0) = \Delta_1 y_1 + \Delta_2 y_2 + \Delta_3 y_1'$$

and

$$f'(0) = \Delta_1' y_1' + \Delta_2' y_2' + \Delta_3' y_1,$$

that is, we have $\prod_{i=1}^{3} U_i^{\Delta_i} = W^{f(0)}$ and $\prod_{j=1}^{3} U_j^{'\Delta_j'} = W^{f'(0)}$. Therefore, we have that $\prod_{i=1}^{3} U_i^{\Delta_i} = \prod_{j=1}^{3} U_j^{'\Delta_j'}$ holds if $M = M'$.

- *Type*-4 *Authorization*: Given $td_4 = H_1(C_2^{\alpha}, C_1, C_2, C_3)$ and $td_4' = \alpha'$, we have:

$$\begin{aligned}C_4 \oplus td_4 &= x_1 \| x_2 \| y_1 \| y_2, \\ C_4' \oplus H_1(C_2^{'td_4'}, C_1', C_2', C_3') &= x_1' \| x_2' \| y_1' \| y_2'.\end{aligned}$$

From above, we have that if $M = M'$, then $\varphi = f(0)$ and $\varphi' = f'(0)$. Hence, we have that $\varphi = \varphi'$ holds if $M = M'$.

3. As for the third condition, we have the following fact:

If $M \neq M'$, then $Pr[f(x) = f'(x)]$ is negligible. So $\Pr[f'(x_1) = y_1]$ (resp. $\Pr[f(x_1') = y_1']$) is negligible, that is, the probability of point (x_1, y_1) (resp. (x_1', y_1')) being on $f'(x)$ (resp. $f(x)$) is negligible. In this case, the probability of $f(0)$ (resp. $f'(0)$) being reconstructed with $(x_1, y_1), (x_2, y_2)$ and (x_1', y_1') (resp. $(x_1', y_1'), (x_2', y_2')$ and (x_1, y_1)) correctly is negligible. That is, $\Pr[\varphi = f(0)]$ (resp. $\Pr[\varphi' = f'(0)]$) is negligible if $M \neq M'$.

From above, the followings hold.

- *Type*-1 *Authorization*: If $Test_1(C, td_1, C', td_1') = 1$, it means that $\varphi = \varphi'$ holds. From above, since $\Pr[\varphi = f(0)]$ and $\Pr[\varphi' = f'(0)]$ are both negligible if $M \neq M'$, we have that $\Pr[Test_1(C, td_1, C', td_1') = 1]$ is negligible.
- *Type*-2 *Authorization*: If $Test_2(C, td_2, C', td_2') = 1$, it means that $\varphi = \varphi'$ holds. From above, since $\Pr[\varphi = f(0)]$ and $\Pr[\varphi' = f'(0)]$ are both negligible if $M \neq M'$, we have that $\Pr[Test_2(C, td_2, C', td_2') = 1]$ is negligible.

- *Type*-3 *Authorization*: If $Test_3(C, td_3, C', td'_3) = 1$, it means that $\prod_{i=1}^{3} V_i^{\Delta_i} = \prod_{j=1}^{3} V_j^{'\Delta'_j}$ holds. From above, since $\Pr[\varphi = f(0)]$ and $\Pr[\varphi' = f'(0)]$ are both negligible if $M \neq M'$, where $\varphi = \Delta_1 y_1 + \Delta_2 y_2 + \Delta_3 y'_1$ and $\varphi' = \Delta'_1 y'_1 + \Delta'_2 y'_2 + \Delta_3 y_1$, we have that $\Pr[Test_3(C, td_3, C',\ td'_3) = 1]$ is negligible.
- *Type*-4 *Authorization*: If $Test_4(C, td_4, C', td'_4) = 1$, it means that $\varphi = \varphi'$ holds. From above, since $\Pr[\varphi = f(0)]$ and $\Pr[\varphi' = f'(0)]$ are both negligible if $M \neq M'$, we have that $\Pr[Test_4(C, td_4, C', td'_4) = 1]$ is negligible. □

5 Security

Theorem 2 *The proposed PKEET-FA scheme is OW-CCA secure for Type-*π *(*$\pi = 1, 2, 3$*) authorization against Type-I adversary (c.f. Definition 3) based on DDH assumption in the random oracle model.*

Proof: Suppose that there is a Type-I adversary $\mathcal{A}_1$ which can break the proposed PKEET-FA scheme, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ which can solve the DDH problem. Let $(g, g^a, g^b, h) \in \mathbb{G}^4$ be an instance of the DDH problem and the target receiver has index t ($1 \leq t \leq n$), $\mathcal{B}$'s task is to check whether or not $g^{ab} = h$ holds. Given the instance, $\mathcal{B}$ and $\mathcal{A}_1$ play the following game.

1. *Setup*: $\mathcal{B}$ runs the algorithm *Setup* to create the system parameters pp, where $q, g, \mathbb{G} \in pp$, and then sends pp to $\mathcal{A}_1$. Then, $\mathcal{B}$ runs the algorithm *KeyGen* to generate n public/private key pairs (pk_i, sk_i) ($1 \leq i \leq n, i \neq t$). We set $pk_t = (X_t, Y_t)$, where $X_t = g^{\alpha_t}$, $Y_t = g^a$ and $\alpha_t \in \mathbb{Z}_q$ is picked randomly. All pk_i are given to $\mathcal{A}_1$. Moreover, lists H-list, H_1-list, $\cdots$, H_7-list, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

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

   - *H-query*(γ): $\mathcal{B}$ picks $\theta \in \{0,1\}^{\lambda+l'}$ randomly, stores a new item $[\gamma, \theta]$ into H-list and returns θ as the answer.
   - H_1-*query*(γ_1, C_1, C_2, C_3): $\mathcal{B}$ picks $\theta_1 \in \{0,1\}^{2l+2l'}$ randomly, stores a new item $[\gamma_1, C_1, C_2, C_3, \theta_1]$ into H_1-list and returns θ_1 as the answer.
   - H_i-*query*(M) ($i = \{2, 4, 6\}$): $\mathcal{B}$ picks $h_i \in \{0,1\}^l$ randomly, stores a new item $[M, h_i]$ into H_i-list and returns h_i as the answer.
   - H_j-*query*(M) ($j = \{3, 5, 7\}$): $\mathcal{B}$ picks $h_j \in \mathbb{Z}_q$ randomly, stores a new item $[M, h_j]$ into H_j-list and returns h_j as the answer.
   - *Key retrieve queries*(i): $\mathcal{B}$ sends $sk_i = (\alpha_i, \beta_i)$ to $\mathcal{A}_1$.
   - *Decryption queries*(i, C): Let $C = (C_1, C_2, C_3, C_4)$.
     - If $i = t$, then for each item $[\gamma, \theta]$ in H-list, $\mathcal{B}$ performs as follows.
       (a) Compute $M \| r_2 = C_3 \oplus \theta$ and $x_1 \| x_2 \| y_1 \| y_2 = C_4 \oplus H_1(C_2^{\alpha_t}, C_1, C_2, C_3)$.
       (b) Compute P_1, P_2 and P_3 as done in the algorithm *Encrypt*.
       (c) Reconstruct $f(x)$ with three points P_1, P_2 and P_3.

(d) If $C_2 = g^{r_2}$, $f(x_1) = y_1$ and $f(x_2) = y_2$ all hold, it returns M to $\mathcal{A}_1$ as the answer. If no such item in H-list, it returns $\perp$ to $\mathcal{A}_1$.

– Else, $\mathcal{B}$ runs algorithm *Decrypt* with C and sk_i as input, and then sends the output of the algorithm to $\mathcal{A}_1$ as the answer.

3. *Authorization queries*: For Type-π ($\pi = 1, 2, 3$) authorization,

   - on input (i), $\mathcal{B}$ returns $td_1 = \alpha_i$;
   - on input (i, C), $\mathcal{B}$ returns $td_2 = H_1(C_2^{\alpha_i}, C_1, C_2, C_3)$, where $C = (C_1, C_2, C_3, C_4)$;
   - on input (i, C, C'), $\mathcal{B}$ returns $td_3 = ([H_1(C_2^{\alpha_i}, C_1, C_2, C_3)]_0^{2l-1}, W^{y_1}, W^{y_2})$, where $W = C_2^{'r_2}$, $C = (C_1, C_2, C_3, C_4)$ and $C' = (C'_1, C'_2, C'_3, C'_4)$.

4. *Challenge*: $\mathcal{B}$ picks a random number $r_2 \in \mathbb{Z}_q$ and a random message $M_t \in \{0,1\}^\lambda$, and then computes $C_t = (C_1, C_2, C_3, C_4)$ as follows.

$$\begin{aligned} C_1 &= g^b \\ C_2 &= g^{r_2} \\ C_3 &= (M_t \| r_2) \oplus H(h) \\ C_4 &= (x_1 \| x_2 \| y_1 \| y_2) \oplus H_1(g^{\alpha_t r_2}, C_1, C_2, C_3), \end{aligned}$$

   where the points (x_1, y_1) and (x_2, y_2) are generated randomly on an interpolating polynomial $f(x)$ with degree 2 which is created by passing through three points: $P_1 = (H_2(M_t), H_3(M_t)), P_2 = (H_4(M_t), H_5(M_t))$ and $P_3 = (H_6(M_t), H_7(M_t))$.

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

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

   (a) (t) does not appear in the key retrieve queries;

   (b) (t, C_t) does not appear in the decryption queries.

6. *Guess*: $\mathcal{A}_1$ outputs M'_t. If $M_t = M'_t$, $\mathcal{B}$ outputs 1 for the challenge instance of the DDH problem, and 0 otherwise. □

Theorem 3 *The proposed PKEET-FA scheme is IND-CCA secure for the Type-π ($\pi = 1, 2, 3$) authorization against Type-II adversary (c.f. Definition 4) based on DDH assumption in the random oracle model.*

Proof: Suppose that there is a Type-II $\mathcal{A}_2$ which can break the proposed PKEET-FA scheme, then we can construct a probabilistic polynomial-time algorithm $\mathcal{B}$ which can solve the DDH problem. Let $(g, g^a, g^b, h) \in \mathbb{G}^4$ be an instance of the DDH problem and the target receiver has index t ($1 \le t \le n$), $\mathcal{B}$'s task is to check whether or not $g^{ab} = h$ holds. Given the instance, $\mathcal{B}$ and $\mathcal{A}_2$ play the following game.

1. *Setup*: $\mathcal{B}$ runs the algorithm *Setup* to create the system parameters pp, and then sends pp to $\mathcal{A}_2$. Then, $\mathcal{B}$ runs the algorithm *KeyGen* to generate n public/private key pairs (pk_i, sk_i) ($1 \le i \le n, i \ne t$). We set $pk_t = (X_t, Y_t)$, where $X_t = g^{\alpha_t}$, $Y_t = g^a$ and $\alpha_t \in \mathbb{Z}_q$ is picked randomly. All pk_i are given to $\mathcal{A}_2$. Moreover, lists H-list, H_1-list, $\cdots$, H_7-list, which are initial empty, are maintained by $\mathcal{B}$ to answer the random oracle queries. If the same input is asked multiple times, the same answer will be returned.

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

   - *H-query*(γ): $\mathcal{B}$ picks $\theta \in \{0,1\}^{\lambda+l'}$ randomly and stores a new item $[\gamma, \theta]$ into H-list.
   - *H_1-query*(γ_1, C_1, C_2, C_3): $\mathcal{B}$ picks $\theta_1 \in \{0,1\}^{2l+2l'}$ randomly and stores a new item $[\gamma_1, C_1, C_2, C_3, \theta_1]$ into H_1-list.
   - *H_i-query*(M) ($i = \{2,4,6\}$): $\mathcal{B}$ picks $h_i \in \{0,1\}^l$ randomly and stores a new item $[M, h_i]$ into H_i-list.
   - *H_j-query*(M) ($j = \{3,5,7\}$): $\mathcal{B}$ picks $h_j \in \mathbb{Z}_q$ randomly and stores a new item $[M, h_j]$ into H_j-list.
   - *Key retrieve queries*(i): $\mathcal{B}$ sends $sk_i = (\alpha_i, \beta_i)$ to $\mathcal{A}_2$.
   - *Decryption queries*(i, C): Let $C = (C_1, C_2, C_3, C_4)$.
     - If $i = t$, then for each item $[\gamma, \theta]$ in H-list, $\mathcal{B}$ performs as follows.
       (a) Compute $M \| r_2 = C_3 \oplus \theta$ and $x_1 \| x_2 \| y_1 \| y_2 = C_4 \oplus H_1(C_2^{\alpha_t}, C_1, C_2, C_3)$.
       (b) Compute P_1, P_2 and P_3 as done in the algorithm *Encrypt*.
       (c) Reconstruct $f(x)$ with three points P_1, P_2 and P_3.
       (d) If $C_2 = g^{r_2}$, $f(x_1) = y_1$ and $f(x_2) = y_2$ all hold, it returns M to $\mathcal{A}_1$ as the answer. If no such item in H-list, it returns $\perp$ to $\mathcal{A}_1$.
     - Else, $\mathcal{B}$ runs algorithm *Decrypt* with C and sk_i as input, and then sends the output of the algorithm to $\mathcal{A}_1$ as the answer.

3. *Authorization queries*: For Type-π ($\pi = 1, 2, 3$) authorization,

   - on input (i), $\mathcal{B}$ returns $td_1 = \alpha_i$;
   - on input (i, C), $\mathcal{B}$ returns $td_2 = H_1(C_2^{\alpha_i}, C_1, C_2, C_3)$;
   - on input (i, C, C'), $\mathcal{B}$ returns $td_3 = ([H_1(C_2^{\alpha_i}, C_1, C_2, C_3)]_0^{2l-1}, W^{y_1}, W^{y_2})$, where $W = C_2'^{r_2}$, $C = (C_1, C_2, C_3, C_4)$ and $C' = (C_1', C_2', C_3', C_4')$.

4. *Challenge*: $\mathcal{A}_2$ submits two equal-length messages $M_0, M_1 \in \{0,1\}^\lambda$, $\mathcal{B}$ picks a random bit $\rho \in \{0,1\}$ and a random number $r_2 \in \mathbb{Z}_q$, then computes $C_t^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ as follows.

$$\begin{aligned} C_1^* &= g^b \\ C_2^* &= g^{r_2} \\ C_3^* &= (M_\rho \| r_2) \oplus H(h) \\ C_4^* &= (x_1 \| x_2 \| y_1 \| y_2) \oplus H_1(g^{\alpha_t r_2}, C_1^*, C_2^*, C_3^*), \end{aligned}$$

   where the points (x_1, y_1) and (x_2, y_2) are generated on an interpolating polynomial $f(x)$ with degree 2 which is created by passing through three points: $P_1 = (H_2(M_\rho), H_3(M_\rho)), P_2 = (H_4(M_\rho), H_5(M_\rho))$ and $P_3 = (H_6(M_\rho), H_7(M_\rho))$.

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

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

   (a) (t) does not appear in the key retrieve queries;
   (b) (t, C_t^*) does not appear in the decryption queries.
   (c) For Type-π ($\pi = 1, 2, 3$) authorization queries,

- $\pi = 1$: (t) does not appear in the authorization queries;
- $\pi = 2$: (t, C_t^*) does not appear in the authorization queries;
- $\pi = 3$: $(t, C_t^*, \cdot)$ does not appear in the authorization queries.

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

6 Efficiency

In this section, we compare the proposed scheme with the state-of-the-art scheme [14]. The efficiency comparison of encryption and decryption is shown in Tab. 1, the efficiency comparison of TYPE-1, TYPE-2, TYPE-3 and TYPE-4 authorizations is shown in Tab. 2. Moreover, we show the efficiency comparison between our proposal and the other existing work [18–21] in Tab. 3.

Let Enc, Dec, Aut, Test denote the computational costs of algorithms for encryption, decryption, authorization and equality test, respectively. Let Exp be a modular exponentiation, Pairing denote a pairing evaluation and Inv denote a modular inverse.

Table 1: Efficiency Comparison of Encryption and Decryption

	Enc	Dec
[14]	6Exp	5Exp
Ours	4Exp+6Inv	3Exp+6Inv

Table 2: Efficiency Comparison of Authorizations

		Aut	Test
TYPE-1	[14]	0	2Pairing+2Exp
	Ours	0	2Exp+6Inv
TYPE-2	[14]	2Exp	2Pairing+2Exp
	Ours	1Exp	6Inv
TYPE-3	[14]	2Pairing+2Exp	2Pairing+2Exp+2Inv
	Ours	4Exp	6Exp+6Inv
TYPE-4	[14]	1Exp	2Pairing+2Exp
	Ours	1Exp	1Exp+6Inv

Table 3: Efficiency Comparison with The Other Existing Work

	Enc	Dec	Aut	Test
[21]	3Exp	3Exp	-	2Pairing
[18, 20]	4Exp	2Exp	3Exp	4Pairing+2Inv
[19]	5Exp	2Exp	0	4Exp
Ours(TYPE-1)	4Exp+6Inv	3Exp+6Inv	0	2Exp+6Inv
Ours(TYPE-2)	4Exp+6Inv	3Exp+6Inv	1Exp	6Inv
Ours(TYPE-3)	4Exp+6Inv	3Exp+6Inv	4Exp	6Exp+6Inv
Ours(TYPE-4)	4Exp+6Inv	3Exp+6Inv	1Exp	1Exp+6Inv

We stress here again that a bilinear pairing costs about five times than a modular exponentiation in a conventional desktop computer [11, 13, 23]. Hence, it is clear that our scheme is (surprisingly) more efficient than the state-of-the-art scheme [14].

Compared with [18, 20, 21], our proposal is more efficient in terms of equality test. Compared with [19], TYPE-1, TYPE-2 and TYPE-4 authorizations in our proposal are more efficient in terms of equality test and encryption, while TYPE-3 requires a bit more in terms of equality test.

As for the security, our proposal achieves OW-CCA security with authorization and IND-CCA security without authorization. Moreover, all the schemes in comparison are proven secure in random oracle model based on standard and widely accepted assumptions.

In conclusion, our scheme supports much more flexible authorization compared with the other existing work [18–21] without sacrificing efficiency. In fact, our scheme enjoys both efficiency and flexibility. Compared with the state-of-the-art scheme [14], our scheme is also more efficient, especially the equality test process. As the basic operation, millions of the equality test process may be performed by the cloud server every day. Thus, our proposal can save huge of computational resources. Hence, compared with all existing work, we believe that our proposal is more practical.

7 Conclusion

In this paper, we present a new PKEET-FA scheme without bilinear pairings by following the definition and security models proposed in [14]. Compared with the state-of-the-art scheme [14], our proposal is more efficient. And compared with the other existing work [18–21], our proposal enjoys both efficiency and flexibility.

Acknowledgment

This work was supported by Shandong Special Project of Education Enrollment Examination (No.ZK1437B005), Chinese Ministry of Education, Humanities and Social Sciences Research Project (No. 14YJCZH136), Shandong "Twelfth Five Year" Language Application Research Project (No.3032), The First Characteristic of Elite Schools Construction Project of Qingdao University (No.05091304), Innovative Teaching Laboratory Research Project in 2014 of Qingdao University (No.10).

References

[1] M. Abdalla et al., "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," J. Cryptol., vol. 21, no. 3, pp. 350-391, Mar. 2008.

[2] M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and efficiently searchable encryption," in Advances in Cryptology. Santa Barbara, CA, USA: Springer-Verlag, 2007, pp. 535-552.

[3] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in Advances in Cryp-

tology (Lecture Notes in Computer Science), vol. 5157. Berlin, Germany: Springer-Verlag, Aug. 2008, pp. 360-378.

[4] A. Boldyreva, S. Fehr, and A. O'Neill, "On notions of security for deterministic encryption, and efficient constructions without random oracles," in Advances in Cryptology. Santa Barbara, CA, USA: Springer-Verlag, 2008, pp. 335-359.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Advances in Cryptology (Lecture Notes in Computer Science), vol. 3027. Berlin, Germany: Springer-Verlag, May 2004, pp. 506-522.

[6] Z. Brakerski and G. Segev, "Better security for deterministic publickey encryption: The auxiliary-input setting," J. Cryptol., vol. 27, no. 2, pp. 210-247, Apr. 2014.

[7] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in Secure Data Management (Lecture Notes in Computer Science), vol. 4165. Berlin, Germany: Springer-Verlag, Sep. 2006, pp. 75-83.

[8] T. Fuhr and P. Paillier, "Decryptable searchable encryption," in Provable Security (Lecture Notes in Computer Science), vol. 4784. Berlin, Germany: Springer-Verlag, Nov. 2007, pp. 228-236.

[9] L. Ibraimi, S. Nikova, P. Hartel, and W. Jonker, "Public-key encryption with delegated search," in Applied Cryptography and Network Security. Berlin, Germany: Springer-Verlag, Jun. 2011, pp. 532-549.

[10] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" Comput. Commun., vol. 32, no. 2, pp. 394-396, Feb. 2009.

[11] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," IEEE Trans. Wireless Commun., vol. 11, no. 1, pp. 62-67, Feb. 2004.

[12] Y. Lu, R. Zhang, and D. Lin, "Stronger security model for public-key encryption with equality test," in Pairing-Based Cryptography-Pairing (Lecture Notes in Computer Science), vol. 7708. Berlin, Germany: Springer-Verlag, May 2012, pp. 65-82.

[13] B. Lynn. Pairing Based Cryptography-Benchmarks. [Online]. Available: http://crypto.stanford.edu/pbc/times.html, Aug. 2014.

[14] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient Public Key Encryption With Equality Test Supporting Flexible Authorization," IEEE Trans. on Information Forensics and Security, vol. 10, no. 3, pp.458-470, 2015.

[15] S. Ma, M. Zhang, Q. Huang, and B. Yang, "Public key encryption with delegated equality test in a multi-user setting," Comput. J., 2015.

[16] M. Nishioka, "Perfect keyword privacy in PEKS systems," in Provable Security. Berlin, Germany: Springer-Verlag, Sep. 2012, pp. 175-192.

[17] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," IEICE Electron. Exp., vol. 6, no. 5, pp. 237-243, 2009.

[18] Q. Tang, "Public key encryption schemes supporting equality test with authorisation of different granularity," Int. J. Appl. Cryptography, vol. 2, no. 4, pp. 304-321, Jul. 2012.

[19] Q. Tang, "Public key encryption supporting plaintext equality test and user-specified authorization," Secur. Commun. Netw., vol. 5, no. 12, pp. 1351-1362, Dec. 2012.

[20] Q. Tang, "Towards public key encryption scheme supporting equality test with fine-grained authorization," in Proc. 16th Austral. Conf. Inf. Secur. Privacy, vol. 6812. Melbourne, Australia, Jul. 2011, pp. 389-406.

[21] G. Yang, C. H. Tan, Q. Huang, and D. S.Wong, "Probabilistic public key encryption with equality test," in Topics in Cryptology, vol. 5985. Berlin, Germany: Springer-Verlag, Mar. 2010, pp. 119-131.

[22] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in Autonomic and Trusted Computing (Lecture Notes in Computer Science), vol. 5060. Berlin, Germany: Springer-Verlag, Jun. 2008, pp. 100-105.

[23] M. Yoshitomi, T. Takagi, S. Kiyomoto, and T. Tanaka, "Efficient implementation of the pairing on mobile phones using BREW," IEICE Trans. Inf. Syst., vol. E91-D, no. 5, pp.1330-1337, May 2008.

Xi-Jun Lin, corresponding author, is lecturer at the Department of Computer Science and Technology, Ocean University of China. He has M.Sc. in Ocean University of China and a Ph.D. in Chinese Academy of Sciences. His research interests include cryptography and information security. email:linxj77@163.com.

Haipeng Qu is associate professor at the Department of Computer Science and Technology, Ocean University of China. He has M.Sc. in Ocean University of China and a Ph.D. in Chinese Academy of Sciences. His research interests include information security and sensor network.

Xiaoshuai Zhang is a Master Degree candidate at the Department of Computer Science and Technology, Ocean University of China. His research interests include cryptography and cyber security.