Towards Public Key Encryption Scheme Supporting Equality Test with Fine-Grained Authorization

Qiang Tang

DIES, Faculty of EEMCS
University of Twente, the Netherlands
q.tang@utwente.nl

Abstract. In this paper we investigate a new category of public key encryption schemes which supports equality test between ciphertexts. With this new primitive, two users, who possess their own public/private key pairs, can issue token(s) to a proxy to authorize it to perform equality test between their ciphertexts. We provide a formulation and a corresponding construction for this primitive, and our formulation provides fine-grained authorization policy enforcements for users. With the increasing popularity of outsourcing data and computations to third-party service providers, this new primitive will be an important building block in designing privacy protection solutions supporting operations on encrypted data.

Keywords: public key encryption, equality test, fine-grained authorization

1 Introduction

Today, more and more IT applications outsource the storage and business transactions of corporate/personal database to third-party service providers. For such applications, it is a big challenge to design mechanisms, which simultaneously achieve the intended business objectives and provide a maximal level of privacy guarantee on the sensitive data. Within the information security community, a lot of research efforts have been dedicated to cryptographic techniques supporting operations on encrypted data. In this paper, we are interested in Public Key Encryption schemes which support Equality Test between ciphertexts. This primitive is formally referred to as PKEET, and an informal functional description is as follows.

Given a public key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, suppose that two users possess their public/private key pairs* (PK, SK) *and* (PK', SK') *respectively. If this public key encryption scheme belongs to the category of PKEET, then the two users can authorize a third-party proxy to perform the following test: Given* $\mathsf{Enc}(M, PK)$ *and* $\mathsf{Enc}(M', PK')$ *for any* M *and* M'*, test whether* $M = M'$ *without knowing* M *or* M'*.*

As mentioned in [20], PKEET is a useful building block in construct privacy-preserving applications, such as outsourced databases. Besides, we can foresee more applications in the emerging computing scenarios. For example, in an Internet-based PHR application [17], a PKEET cryptosystem can allow patients to encrypt their attributes and a semi-trusted proxy to match the encrypted attributes and recommend the patients to each other.

1.1 Related Work

The concept of PKEET cryptosystem was proposed by Yang *et al.* [20]. However, their formulation lacks an authorization mechanism for users to specify who can perform equality test between their ciphertexts, and in fact any entity can perform the equality test. The consequence is that standard semantic security or IND-CPA security cannot be achieved against any entity, when considering the fact that ciphertexts are public information. In addition, if the message space is polynomial size or the min-entropy of the message distribution is much lower than the security parameter, then any entity can potentially mount an offline message recovery attack. This attack is similar to the offline keyword guessing attack in the case of PEKS (or searchable encryption) [11,18].

The concepts of PKEET has a close nature to that of Public key encryption with keyword search (PEKS) [8] and public key encryption with registered keyword search (PERKS) [18]. With a PEKS or PERKS scheme, a user can enable a server to perform equality test between the keywords embedding in a tag and a ciphertext, and the user enforces her authorization by issuing a token to the server. The difference is that, instead of keywords, PKEET is concerned with the equality test of plaintexts which are encrypted under different public keys. Another related concept is order preserving encryption (OPE) scheme, which is a primitive firstly proposed by Agrawal *et al.* [1] and then further investigated by Boldyreva *et al.* [6]. With an OPE scheme, the order of ciphertexts always remains the same as that of the corresponding plaintexts. Therefore, given a set of ciphertexts, any entity can directly compare the plaintexts. The order-preserving property of an OPE scheme holds only for the ciphertexts generated under the same public key, which differs from the purpose of PKEET.

1.2 Our Contribution

To mitigate the potential vulnerabilities of PKEET, we integrate a fine-grained authorization policy enforcement mechanism into PKEET and propose an enhanced primitive, namely FG-PKEET. With an FG-PKEET cryptosystem, two users, say Alice and Bob, need to run the authorization algorithm together to issue a token to a semi-trusted proxy, which will then be authorized to perform equality test between their ciphertexts. Without the token, the equality test cannot be performed. With this new primitive, users gain more control over the operations on their encrypted data.

– A user has tight control over who can perform equality test on her ciphertexts, by choosing the semi-trusted proxies.
– A user has tight control over with whose ciphertexts that her ciphertexts can be tested with, by choosing with which user to run the authorization algorithm.

For FG-PKEET, we consider two types of adversaries: Type-I adversary which represents the semi-trusted proxies, and Type-II adversary which represents all malicious entities. With respect to a Type-I adversary, we provide OW-CCA (i.e. one-way CCA) and OW-CPA (i.e. one-way CPA) security definitions; while with respect to a Type-II adversary, we provide standard IND-CCA and IND-CPA security definitions. Furthermore, a fine-grained authorization property is defined for FG-PKEET. Informally, this property means that a proxy cannot perform equality test between two users' ciphertexts unless it receives a token assigned by these two users together. For example, a proxy cannot compare the ciphertexts of Bob and Charlie, even if it has received a token to compare the ciphertexts of Alice and Bob together with another token to compare the ciphertexts of Alice and Charlie. We propose an FG-PKEET cryptosystem, which achieves all the security properties defined in our security model.

In the extreme situation, when the message space is polynomial size or the min-entropy of the message distribution is much lower than the security parameter, for FG-PKEET, only a Type-I adversary is capable of mounting an offline message recovery attack which is unavoidable due to the desired equality test functionality. However, compared with the formulation in [20], where any adversary can mount the attack, our formulation achieves a significant security improvement. Furthermore, based on computational client puzzles [14], we propose an enhancement to mitigate this type of attack.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we formulate the concept of FG-PKEET. In Section 3, we propose an FG-PKEET cryptosystem. In Section 4, we analyse the proposed cryptosystem and provide an enhancement. In Section 5, we conclude the paper.

2 Formulation of FG-PKEET

In this section, we first provide a formal description for FG-PKEET, and then present the security model.

Throughout the paper, we use "$\|$" to denote the concatenation operator and use $x \in_R X$ to denote that x is chosen from X uniformly at random.

2.1 Description of FG-PKEET

An FG-PKEET cryptosystem consists of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aut}, \mathsf{Com})$, where $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ define a standard public key encryption scheme while $(\mathsf{Aut}, \mathsf{Com})$ define the equality test functionality.

- $\mathsf{KeyGen}(\ell)$: This algorithm takes a security parameter ℓ as input, and outputs a public/private key pair (PK, SK). Let $\mathcal{M}$ denote the message space.
- $\mathsf{Enc}(M, PK)$: This algorithm takes a message $M \in \mathcal{M}$ and the public key PK as input, and outputs a ciphertext C.
- $\mathsf{Dec}(C, SK)$: This algorithm takes a ciphertext C and the private key SK as input, and outputs the plaintext M or an error message $\perp$.

Let all the potential users be denoted as U_i $(1 \leq i \leq N)$, where N is an integer, and they adopt the above public key encryption scheme. For any i, suppose that U_i's key pair is denoted as (PK_i, SK_i). Suppose that U_i and U_j want to enable a proxy to perform equality test between their ciphertexts, the Aut and Com algorithms are defined as follows.

- $\mathsf{Aut}(SK_i; SK_j; \cdot)$: This algorithm is interactively run among U_i, U_j and the proxy, and the two users use their private keys as their secret inputs. At the end of the algorithm execution, the proxy receives a token $T_{i,j}$ as the output, while U_i and U_j receive no explicit output.
- $\mathsf{Com}(C_i, C_j, T_{i,j})$: This algorithm takes two ciphertexts C_i, C_j and the token $T_{i,j}$ as input, and outputs 1 if $M_i = M_j$ or 0 otherwise. Note that C_i, C_j are two ciphertexts encrypted under PK_i and PK_j respectively.

In the algorithm definitions, besides the explicitly specified parameters, other public parameters could also be specified and be implicitly part of the input. We omit those parameters for the simplicity of description. Note that, under our definition of Aut, $T_{i,j}$ and $T_{j,i}$ are exactly the same thing.

It is worth noting that the Aut algorithm is supposed to run interactively among two users and the proxy. The interactive nature of this algorithm may seem to be a drawback, but it in fact reflects the process that the two users together authorize the semi-trusted proxy to perform equality test between their ciphertexts. Moreover, this algorithm only needs to be run once for any selected proxy, which will then be able to compare all ciphertexts of the two users. Therefore, the interactive nature of the the Aut algorithm will not be a performance bottleneck in practice.

Similar to other cryptographic primitives, the basic requirement for FG-PKEET is soundness. Informally, this property means that the algorithms Dec and Com work properly with valid inputs. Formally, it is defined as follows.

Definition 1. *An FG-PKEET cryptosystem achieves (unconditional) soundness if the following two equalities hold for any $i, j \geq 1$ and $M, M' \in \mathcal{M}$. Let $(PK_i, SK_i) = \mathsf{KeyGen}(\ell)$ and $(PK_j, SK_j) = \mathsf{KeyGen}(\ell)$.*

1. $\mathsf{Dec}(\mathsf{Enc}(M, PK_i), SK_i) = M$ *and* $\mathsf{Dec}(\mathsf{Enc}(M', PK_j), SK_j) = M'$.
2. $\mathsf{Com}(\mathsf{Enc}(M, PK_i), \mathsf{Enc}(M', PK_j), \mathsf{Aut}(SK_i; SK_j; \cdot))$ *is equal to 1 if $M = M'$, and 0 otherwise.*

As a remark, in the definitions of Aut and Com, we implicitly assume that $i \neq j$ because we are only interested in testing the equality of the ciphertexts of two different users.

2.2 The Security Model

To facilitate our formal discussions, we make the following assumptions.

1. First of all, all users honestly generate their public/private key pairs and the execution of the Aut algorithm will be carried out through secure channels between the involved entities.
2. Secondly, the proxies are semi-trusted (or, honest-but-curious) to the users who have chosen them. They will faithfully follow the protocol specifications, but will try to deduce some information from the acquired data. In addition, one proxy can serve multiple pairs of users to perform equality test.

3. Thirdly, there is no overlap between the user set and the proxy set, namely no user will be allowed to act as a proxy for another two users. This will greatly simplify our discussion. Yet, we leave it as a future work to investigate FG-PKEET in the case where this assumption is not true.

With respect to an FG-PKEET cryptosystem, for an honest user U_t, where $1 \leq t \leq N$, we consider two categories of adversaries, namely Type-I and Type-II adversaries as illustrated in Fig. 1.

1. Type-I adversary represents the semi-trusted proxies with which U_t has run the algorithm Aut with. Referring to Fig. 1, Proxy I and Proxy L are Type-I adversary.
2. Type-II adversary represents all possibly malicious entities in the system from the perspective of U_t, namely U_i $(1 \leq i \leq N, i \neq t)$. In fact, all proxies with which U_t has not run the algorithm Aut should also be regarded as a malicious adversary, because U_t do not even semi-trust them. For example, Proxy T in Fig. 1 is such an entity. However, taking them into account will not give the Type-II adversary extra power, so that we simply ignore them.

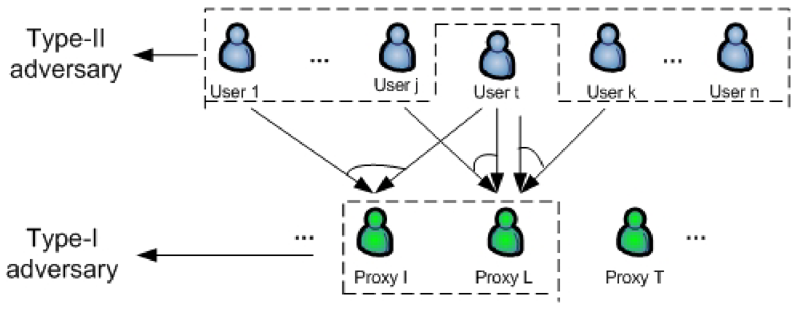

Fig. 1. An Illustration of Adversaries for FG-PKEET

As to a Type-I adversary, it is involved in the executions of the Aut algorithm as the proxy with U_t, and obtains the tokens, and it may also obtain some information about U_t's plaintexts through accessing U_t's decryption oracle. Clearly, in the presence of a Type-I adversary, standard indistinguishability notions, such as IND-CCA and IND-CPA, cannot be achieved[1]. Against a Type-I adversary, we consider the following two security properties.

1. OW-CCA (i.e. one-wayness under a chosen ciphertext attack), which implies that an adversary cannot recover the plaintext from a ciphertext $C_t^* =$

[1] Referring to Fig. 1, given $\mathsf{Enc}(M_t, PK_t)$, Proxy L is able to test whether M_t is equal to any M. Since the proxy has been authorized by U_t and U_k together, to do so, it just needs to run a test between $\mathsf{Enc}(M_t, PK_t)$ and $\mathsf{Enc}(M, PK_k)$.

$\mathsf{Enc}(M_t, PK_t)$ even if it is allowed to query the decryption oracle with any ciphertext except for C_t^*. This is the best achievable security guarantee considering the desired equality test functionality.

2. Fine-grained authorization property, which means that if two users have not authorized a proxy to perform equality test between their ciphertexts then the proxy should not be able to do so. Referring to Fig. 1, U_t and U_n have not authorized Proxy L to perform equality test between their ciphertexts, so that it should not be able to do so even if U_t has authorized it to perform equality test between her ciphertexts and those of U_j and U_k. It is worth noting this is an analog to the collusion resistance property in the attribute-based encryption schemes [15].

As to the power of a Type-II adversary, it is involved in the executions of the Aut algorithm as the other user with U_t, so that it may learn some information about U_t's private key. Moreover, it may also obtain some information about U_t's plaintexts through accessing U_t's decryption oracle. In the presence of a Type-II adversary, we define the standard IND-CCA security.

Note that it is straightforward to define the CPA security by simply disallowing the adversary's access to the Dec oracle in the attack games, so that we omit the details in this paper.

2.3 OW-CCA Security against a Type-I Adversary

Definition 2. *An FG-PKEET cryptosystem achieves OW-CCA security against a Type-I adversary, if, for any $1 \leq t \leq N$, any polynomial-time adversary has only a negligible advantage in the attack game shown in Fig. 2, where the advantage is defined to be* $\Pr[M_t' = M_t]$.

1. The challenger runs KeyGen to generate public/private key pairs (PK_i, SK_i) for all $1 \leq i \leq N$.
2. Phase 1: The adversary is allowed to issue the following types of oracle queries.
   (a) Dec query with data C as input for the index i: the challenger returns $\mathsf{Dec}(C, SK_i)$.
   (b) Aut query with two integer indexes i, j as input: the challenger runs the Aut algorithm with the adversary which plays the role of the proxy.

   At some point, the adversary asks the challenger for a challenge for an index t.
3. Challenge phase: The challenger chooses a message $M_t \in_R \mathcal{M}$ and sends $C_t^* = \mathsf{Enc}(M_t, PK_t)$ to the adversary.
4. Phase 2: The adversary is allowed to issue the same types of oracle queries as in Phase 1. In this phase, the adversary's activities should adhere to the following restriction: *The Dec oracle should not have been queried with the data C_t^** for the index t.
   At some point, the adversary terminates by outputting a guess M_t'.

Fig. 2. The Game for OW-CCA

It is worth noting that, strictly speaking, the notion of OW-CCA is neither weaker nor stronger than IND-CPA [3]. One one hand, an IND-CPA secure scheme may not be OW-CCA. For instance, many homomorphic encryption schemes, such as Elgamal [12] and Paillier scheme [13], are IND-CPA but they are clearly not OW-CCA. On the other hand, an OW-CCA secure scheme may not be IND-CPA. For instance, the scheme proposed in Section 3 is OW-CCA but it is not IND-CPA.

2.4 Fine-grained authorization property

Definition 3. *An FG-PKEET cryptosystem achieves the fine-grained authorization property against a Type-I adversary, if, for any $1 \leq t \leq N$, any polynomial-time adversary has only a negligible advantage in the attack game shown in Fig. 3, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. The challenger runs KeyGen to generate public/private key pairs (PK_i, SK_i) for all $1 \leq t \leq N$.
2. Phase 1: The adversary is allowed to issue the following types of oracle queries.
   (a) Dec query with data C as input for the index i: the challenger returns $\mathsf{Dec}(C, SK_i)$.
   (b) Aut query with two integer indexes i, j as input: the challenger runs the Aut algorithm with the adversary which plays the role of the proxy.

   At some point, the adversary sends two integer indexes t, w to the challenger for a challenge. In this phase, the Aut oracle should not have been queried with two integer indexes t, w.
3. Challenge phase: The challenger randomly chooses two different messages M_0, M_1 from $\mathcal{M}$ and a random bit b. If $b = 0$, send $C_t^* = \mathsf{Enc}(M_0, PK_t)$ and $C_w^* = \mathsf{Enc}(M_0, PK_w)$ to the adversary, otherwise send $C_t^* = \mathsf{Enc}(M_0, PK_t)$ and $C_w^* = \mathsf{Enc}(M_1, PK_w)$.
4. Phase 2: The adversary is allowed to issue the same types of oracle queries as in Phase 1. In this phase, the adversary's activities should adhere to the restriction described in Phase 1, together with the following one: *The* Dec *oracle should not have been queried with the data C_t^* and index t or with the data C_w^* and index w.* At some point, the adversary terminates by outputting a guess b'.

Fig. 3. The Game for the Fine-grained Authorization Property

In the attack game, it is clear that $b = 0$ ($b = 1$) implies the challenge ciphertexts do (not) contain the same plaintext. As a result, the adversary's ability of determining b is equivalent to determining the equality of ciphertexts of U_t and U_w. The adversary is not allowed to access $T_{t,w}$ because we assume the adversary is not authorized by U_t and U_w to perform the equality test.

Note the fact that a FG-PKEET cryptosystem can only achieve OW-CCA but not IND-CPA or IND-CCA. If the adversary is allowed to choose M_0, M_1 in the game, then it can trivially win the game. Therefore, different from a typical IND (indistinguishability) security definition, where the adversary is allowed to choose M_0, M_1, in this game the challenger chooses both messages.

2.5 IND-CCA Security against a Type-II Adversary

Definition 4. *An FG-PKEET cryptosystem achieves IND-CCA security against a Type-II adversary, if, for any $1 \le t \le N$, any polynomial-time adversary has only a negligible advantage in the attack game shown in Fig. 4, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

1. The challenger runs KeyGen to generate public/private key pairs (PK_i, SK_i) for all $1 \le t \le N$.
2. Phase 1: The adversary is allowed to issue the following types of oracle queries.
   (a) $\mathsf{KeyRetrieve}$ query with an integer index i as input: the challenger returns SK_i to the adversary.
   (b) Dec query with data C as input for the index i: the challenger returns $\mathsf{Dec}(C, SK_i)$.
   (c) Aut query, defined as below.

   At some point, the adversary sends an integer index t and two messages M_0, M_1 from $\mathcal{M}$ to the challenger for a challenge. In this phase, the adversary's activities should adhere to the following criteria.
   (a) The $\mathsf{KeyRetrieve}$ oracle should not have been queried with the index t.
   (b) For any $i \neq t$, the adversary is allowed to issue Aut oracle queries with indexes i, t as input, for any $i \neq t$, where the adversary plays the role of U_i.
3. Challenge phase: The challenger selects $b \in_R \{0, 1\}$ and sends $C_t^* = \mathsf{Enc}(M_b, PK_t)$ to the adversary.
4. Phase 2: The adversary is allowed to issue the same types of oracle queries as in Phase 1. In this phase, the adversary's activities are subject to the restrictions described in Phase 1, together with the following one: *The* Dec *oracle should not have been queried with the data C_t^* and index t.* At some point, the adversary terminates by outputting a guess b'.

Fig. 4. The Game for IND-CCA

In this game, the challenger generates all key pairs while the adversary is allowed to adaptively retrieve all private keys except SK_t. This formulation faithfully describe the power of a Type-II adversary in our security model, as defined in Section 2.2. In particular, the adversary is allowed to issue Aut oracle queries, which reflects the fact that U_t may interactively run the Aut algorithm with a Type-II adversary. A PKEET is IND-CCA secure against a Type-II adversary implies that, for U_t, the execution of the Aut algorithm leaks no information to other users.

3 A New FG-PKEET Cryptosystem

The proposed cryptosystem has $(\ell, \mathbb{G}, g, p, \mathsf{H}_1, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, q, \mathsf{H}_2, \mathsf{H}_3)$ as the global parameters which are defined as follows.

1. ℓ is the security parameter, $\mathbb{G}$ is a multiplicative group of prime order p, g is a generator of $\mathbb{G}$, and $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^\ell$ is a cryptographic hash function.

2. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative groups of prime order q, and they have g_1 and g_2 as their generators respectively. $\mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{m+d_1}$, $\mathsf{H}_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$ are two cryptographic hash functions, where m is a polynomial in ℓ, $\{0,1\}^m$ is the message space and d_1 is the bit-length of p.

Note the fact that, in a PKEET cryptosystem, a ciphertext allows the receiver to decrypt and also allows a proxy to perform equality test. Hence, the intuition behind our construction is to integrate some extra components into a standard public key encryption scheme, so that these components will facilitate the equality test functionality. Specifically, in the encryption algorithm of the proposed scheme described in next subsection, the extra components are $C^{(2)}$ and $C^{(4)}$.

3.1 The Public key Encryption Scheme

With the above global parameters defined, we first define the public key encryption algorithms ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$).

- $\mathsf{KeyGen}(\ell)$: This algorithm outputs a private key $SK = (x, y)$, where $x \in_R \mathbb{Z}_p$ and $y \in_R \mathbb{Z}_q$, and the corresponding public key is $PK = (g^x, g_1^y)$. Note that the message space is $\mathcal{M} = \{0,1\}^m$.
- $\mathsf{Enc}(M, PK)$: This algorithm outputs a ciphertext $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, where
$$u \in_R \mathbb{Z}_p,\ C^{(1)} = g^u,\ C^{(3)} = \mathsf{H}_2(g^{ux}) \oplus M||u,\ v \in_R \mathbb{Z}_q,$$
$$C^{(2)} = g_1^v,\ C^{(4)} = g_1^{vy} \cdot \mathsf{H}_3(M),\ C^{(5)} = \mathsf{H}_1(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u).$$
- $\mathsf{Dec}(C, SK)$: This algorithm first computes $M'||u' = C^{(3)} \oplus \mathsf{H}_2((C^{(1)})^x)$, and then check the following
  1. $g^{u'} = C^{(1)}$,
  2. $\mathsf{H}_1(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u') = C^{(5)}$.

  If all checks pass, output M', otherwise output an error message $\perp$.

Suppose that every user U_i, for $1 \leq t \leq N$, adopts the above public key encryption scheme. To facilitate our description, we use the index i for all the variables in defining U_i's data. For example, U_i's key pair is denoted as (PK_i, SK_i), where $SK = (x_i, y_i)$ and $PK = (g^{x_i}, g_1^{y_i})$, and U_i's ciphertext $C_i = (C_i^{(1)}, C_i^{(2)}, C_i^{(3)}, C_i^{(4)}, C_i^{(5)})$ is written in the following form.

$$u_i \in_R \mathbb{Z}_p,\ C_i^{(1)} = g^{u_i},\ C_i^{(3)} = \mathsf{H}_2(g^{u_i x_i}) \oplus M_i||u_i,\ v_i \in_R \mathbb{Z}_q,$$

$$C_i^{(2)} = g_1^{v_i},\ C_i^{(4)} = g_1^{v_i y_i} \cdot \mathsf{H}_3(M_i),\ C_i^{(5)} = \mathsf{H}_1(C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i).$$

3.2 The Token Generation Algorithm

Suppose that U_i and U_j want a proxy to perform equality test between their ciphertexts, then they run the following Aut algorithm to generate the token $T_{i,j}$ for the proxy.

– $\mathsf{Aut}(SK_i, SK_j, \cdot)$: This algorithm results in a token $T_{i,j} = (g_2^{r_{i,j}}, g_2^{y_i r_{i,j}}, g_2^{y_j r_{i,j}})$ for the proxy. In more details, the token is interactively generated as follows.
  1. U_i and U_j generate $r_{i,j} \in_R \mathbb{Z}_q$ together.
  2. U_i sends $g_2^{r_{i,j}}, g_2^{y_i r_{i,j}}$ to the proxy, and U_j sends $g_2^{y_j r_{i,j}}$ to the proxy.

Note that, there can be many different ways for U_i and U_j to generate $r_{i,j}$ in implementing this algorithm. For instance, they can use a interactive coin flipping protocol, such as that of Blum [5]. Or, simply they can exchanges two nonces and set $r_{i,j}$ to be the hash value of them. In addition, the security properties will not be affected if U_j is required to send $g_2^{r_{i,j}}$ to the proxy.

3.3 The Equality Test Algorithm

Suppose a proxy has received the token $T_{i,j}$, then it can run the following Com algorithm to perform equality test between the ciphetexts C_i and C_j, which are encrypted under PK_i and PK_j respectively.

– $\mathsf{Com}(C_i, C_j, T_{i,j})$: This algorithm outputs 1 if $x_i = x_j$ or 0 otherwise, where

$$x_i = \frac{\hat{e}(C_i^{(4)}, g_2^{r_{i,j}})}{\hat{e}(C_i^{(2)}, g_2^{y_i r_{i,j}})} = \frac{\hat{e}(g_1^{v_i y_i} \cdot \mathsf{H}_3(M_i), g_2^{r_{i,j}})}{\hat{e}(g_1^{v_i}, g_2^{y_i r_{i,j}})} = \hat{e}(\mathsf{H}_3(M_i), g_2)^{r_{i,j}}$$

$$x_j = \frac{\hat{e}(C_j^{(4)}, g_2^{r_{i,j}})}{\hat{e}(C_j^{(2)}, g_2^{y_j r_{i,j}})} = \frac{\hat{e}(g_1^{v_j y_j} \cdot \mathsf{H}_3(M_j), g_2^{r_{i,j}})}{\hat{e}(g_1^{v_j}, g_2^{y_j r_{i,j}})} = \hat{e}(\mathsf{H}_3(M_j), g_2)^{r_{i,j}}$$

In this construction, the group $\mathbb{G}$ can be any multiplicative group which holds the CDH assumption. In face, it can be set to be $\mathbb{G}_1$ or $\mathbb{G}_2$, in which case $p = q$. We keep it the present way for a general construction.

In Section 2, we stated that we are only interested in testing the equality of the ciphertexts of two different users. For the proposed cryptosystem, the token $T_{i,j}$ actually allows the proxy to perform equality test between the ciphetrexts of U_i (and also U_j). On one hand, this can be regarded as an extra functionality of the cryptosystem. On the other hand, people may argue that this is a potential vulnerability. We leave it as a future work to address this.

4 Comprehensive Security Analysis

In this section, we first prove that the proposed cryptosystem in Section 3 is secure in our security model. Then, we show how to improve its security against a Type-I adversary.

4.1 Preliminary

Following the work by Bellare and Rogaway [4], we use random oracle to model hash functions in our security analysis. A function $P(k) : \mathbb{Z} \rightarrow \mathbb{R}$ is said to be negligible with respect to k if, for every polynomial $f(k)$, there exists an integer N_f such that $P(k) < \frac{1}{f(k)}$ for all $k \geq N_f$.

We say that the CDH (computational Diffie-Hellman) assumption holds in $\mathbb{G}$ of prime order p if, given g^a, g^b where g is a group generator and $a, b \in_R \mathbb{Z}_p$, an adversary has only a negligible advantage in computing g^{ab}.

We say that the DDH (decisional Diffie-Hellman) assumption holds in $\mathbb{G}_1$ of prime order q, if an adversary has only a negligible advantage in distinguishing (g_1^a, g_1^b, g_1^{ab}) from (g_1^a, g_1^b, g_1^c) where g_1 is a group generator and $a_1, b_1, c_1 \in_R \mathbb{Z}_q$. In the pairing setting, namely there is an efficient and non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, the DDH assumption in $\mathbb{G}_1$ is also referred to as the XDH (external Diffie-Hellman) assumption [7].

In order to prove the fine-grained authorization property, we need a new assumption, referred to as extended DBDH (decisional bilinear Diffie-Hellman) assumption. Let a pairing setting be $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where the order of groups is a prime q. The extended DBDH problem is formulated as follows.

1. The challenger selects $g_1, g_4, g_5 \in_R \mathbb{G}_1$, and $g_2, g_3 \in_R \mathbb{G}_2$, and $x_1, y_1, \in_R \mathbb{Z}_p$, and $\alpha, \beta \in_R \mathbb{G}_1$. The challenger flips a coin $b \in_R \{0, 1\}$ and sends X_b to the adversary, where

$$X_0 = (g_1^{x_1},\ g_2^{x_1},\ g_4^{x_1} \cdot \alpha,\ g_1^{y_1},\ g_3^{y_1},\ g_5^{y_1} \cdot \alpha)$$

$$X_1 = (g_1^{x_1},\ g_2^{x_1},\ g_4^{x_1} \cdot \alpha,\ g_1^{y_1},\ g_3^{y_1},\ g_5^{y_1} \cdot \beta)$$

2. The adversary's outputs a guess b'. The adversary's advantage is $|\Pr[b = b'] - \frac{1}{2}|$.

The extended DBDH problem is at most as hard as the XDH problem in a Type-3 pairing setting [10]. In other words, if there is an algorithm to solve the XDH problem then there must be an algorithm to solve the extended DBDH problem, but it is not clear whether the vise-versa is true. Nonetheless, similar to the proof of the implicit XDH assumption in [2], we can show the extended DBDH assumption is hard in the generic group model. We leave the details to the full paper.

4.2 Proof Results

It is straightforward to verify that the soundness property is achieved, namely the Dec and Com work properly. We skip the details here.

Theorem 1. *The proposed FG-PKEET cryptosystem is OW-CCA secure against a Type-I adversary in the random oracle model based on the CDH assumption in* $\mathbb{G}$.

Proof sketch of Theorem 1. Suppose an adversary has the advantage ϵ in the attack game shown in Fig. 2. The security proof is done through a sequence of games [16].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from the adversary, and all hash functions are treated as random oracles. Let $\epsilon_0 = \Pr[M'_t = M_t]$. Clearly, $\epsilon_0 = \epsilon$ holds.

Game_1: In this game, the challenger performs identically to that in Game_0 except that the following. For any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger computes $M_i \| u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)}$ and verifies $g^{u_i} = C_i^{(1)}$. If the verification fails, return $\perp$. Then, the challenger checks whether there exists an input query $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i)$ to H_1, which outputs $C_i^{(5)}$. If such an input query exists, return M_i; otherwise return $\perp$. Let the event Ent_1 be that, for some C_i, a fresh input $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i$ to H_1 results in $C_i^{(5)}$. Clearly, This game is identical to Game_0 unless the event Ent_1 occurs. It is straightforward that $\Pr[Ent_1]$ is negligible if H_1 is modeled as a random oracle. Let $\epsilon_1 = \Pr[M'_t = M_t]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_1 - \epsilon_0| \le \Pr[Ent_1]$.

Game_2: In this game, the challenger performs identically to that in Game_1 except that, for any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger does the following. Try to obtain the query to the oracle H_1 with the input $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i$ satisfying

$$M_i \| u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)},\ g^{u_i} = C_i^{(1)},\ \mathsf{H}_1(C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i) = C_i^{(5)}.$$

If such a query cannot be found, return $\perp$. Otherwise, return M_i. This game is indeed identical to Game_1. Let $\epsilon_2 = \Pr[M'_t = M_t]$, then we have $\epsilon_2 = \epsilon_1$.

Game_3: In this game, the challenger performs identically to that in Game_2 except that the challenge C_t^* is generated as follows.

$$C_t^{(1)} = g^{u_t},\ C_t^{(2)} = g_1^{v_t}, \delta \in_R \{0,1\}^{m+d_1},\ C_t^{(3)} = \delta,$$

$$C_t^{(4)} = g_1^{v_t y_t} \cdot \mathsf{H}_3(M_t),\ C_t^{(5)} = \mathsf{H}_1(C_t^{(1)} \| C_t^{(2)} \| C_t^{(3)} \| C_t^{(4)} \| M_t \| u_t).$$

This game is identical to Game_2 unless the event Ent_2 occurs, namely $g^{u_t x_t}$ is queried to the random oracle H_2. Note that the private key x_t is never used to answer the adversary's queries. Therefore, $\Pr[Ent_2]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_3 = \Pr[M'_t = M_t]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_3 - \epsilon_2| \le \Pr[Ent_2]$.

Since H_1 and H_3 are modeled as random oracles, it is clear that ϵ_3 is negligible. From the above analysis, we have that $\epsilon \le \Pr[Ent_1] + \Pr[Ent_2] + \epsilon_3$, which is negligible in the random oracle model based on the CDH assumption in $\mathbb{G}$. The theorem now follows. □

Theorem 2. *The proposed FG-PKEET cryptosystem achieves fine-grained authorization property against a Type-I adversary in the random oracle model based on the CDH assumption in $\mathbb{G}$ and the extended DBDH assumption.*

Proof sketch of Theorem 2. Suppose an adversary has the advantage ϵ in the attack game shown in Fig. 3. The security proof is done through a sequence of games [16].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from the adversary, and all hash functions are treated as random oracles. Let $\epsilon_0 = \Pr[b' = b]$. Clearly, $\epsilon_0 = \epsilon$ holds.

Game_1: In this game, the challenger performs identically to that in Game_0 except that the following. For any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger computes $M_i \| u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)}$ and verifies $g^{u_i} = C_i^{(1)}$. If the verification fails, return $\perp$. Then, the challenger checks whether there exists an input query $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i$ to H_1, which outputs $C_i^{(5)}$. If such an input query exists, return M_i; otherwise return $\perp$. Let the event Ent_1 be that, for some C_i, a fresh input $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i$ to H_1 results in $C_i^{(5)}$. Clearly, This game is identical to Game_0 unless the event Ent_1 occurs. it is straightforward that $\Pr[Ent_1]$ is negligible if H_1 is modeled as a random oracle. Let $\epsilon_1 = \Pr[b' = b]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_1 - \epsilon_0| \leq \Pr[Ent_1]$.

Game_2: In this game, the challenger performs identically to that in Game_1 except that, for any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger does the following. Try to obtain the query to the oracle H_1 with the input $C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i$ satisfying

$$M_i \| u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)},\ g^{u_i} = C_i^{(1)},\ \mathsf{H}_1(C_i^{(1)} \| C_i^{(2)} \| C_i^{(3)} \| C_i^{(4)} \| M_i \| u_i) = C_i^{(5)}.$$

If such a query cannot be found, return $\perp$. Otherwise, return M_i. This game is indeed identical to Game_1. Let $\epsilon_2 = \Pr[b' = b]$, then we have $\epsilon_2 = \epsilon_1$.

Game_3: In this game, the challenger performs identically to that as in Game_2 except the following. The challenge C_t^* is generated as follows.

$$C_t^{(1)} = g^{u_t},\ C_t^{(2)} = g_1^{v_t}, \delta_t \in_R \{0,1\}^{m+d_1},\ C_t^{(3)} = \delta_t,$$

$$C_t^{(4)} = g_1^{v_t y_t} \cdot \mathsf{H}_3(M_0),\ C_t^{(5)} = \mathsf{H}_1(C_t^{(1)} \| C_t^{(2)} \| C_t^{(3)} \| C_t^{(4)} \| M_t \| u_t).$$

The challenge C_w^* is generated as follows.

$$C_w^{(1)} = g^{u_w},\ C_w^{(2)} = g_1^{v_w}, \delta_w \in_R \{0,1\}^{m+d_1},\ C_w^{(3)} = \delta_w,$$

$$C_w^{(4)} = g_1^{v_w y_w} \cdot \mathsf{H}_3(M_b),\ C_w^{(5)} = \mathsf{H}_1(C_w^{(1)} \| C_w^{(2)} \| C_w^{(3)} \| C_w^{(4)} \| M_b \| u_w).$$

This game is identical to Game_2 unless the event Ent_2 occurs, namely $g^{u_t x_t}$ or $g^{u_w x_w}$ is queried to the random oracle H_2. Note that the private keys x_t, x_w are never used to answer the adversary's queries. Therefore, $\Pr[Ent_2]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_3 = \Pr[b' = b]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_3 - \epsilon_2| \leq \Pr[Ent_2]$.

Game_4: In this game, the challenger performs identically to that as in Game_2 except for answering the Aut queries. For U_t and U_w, the challenger chooses $h_i, h_w \in_R \mathbb{Z}_q$ at the beginning of the game. On receiving an Aut query with the inputs i, t, the challenger returns $(g_2^{h_i r}, g_2^{h_i y_i r}, g_2^{h_i y_j r})$, where $r \in_R \mathbb{Z}_q$, and does something similar to answering the query with the input i, w. Let $\epsilon_4 = \Pr[b' = b]$ in this game. It is clear that this game is identical to Game_3, therefore $\epsilon_4 = \epsilon_3$ holds.

Game_5: In this game, the challenger performs identically to that in Game_4 except the following. The challenge C_t^* is generated as follows.

$$C_t^{(1)} = g^{u_t},\ C_t^{(2)} = g_1^{v_t}, \delta_t \in_R \{0,1\}^{m+d_1},\ C_t^{(3)} = \delta_t,$$

$$k_t \in_R \mathbb{Z}_q,\ C_t^{(4)} = g_1^{v_t y_t k_t},\ C_t^{(5)} = \mathsf{H}_1(C_t^{(1)} \| C_t^{(2)} \| C_t^{(3)} \| C_t^{(4)} \| M_t \| u_t).$$

The challenge C_w^* is generated as follows.

$$C_w^{(1)} = g^{u_w},\ C_w^{(2)} = g_1^{v_w}, \delta_w \in_R \{0,1\}^{m+d_1},\ C_w^{(3)} = \delta_w,$$

$$C_w^{(4)} = g_1^{v_w y_w X},\ C_w^{(5)} = \mathsf{H}_1(C_w^{(1)} \| C_w^{(2)} \| C_w^{(3)} \| C_w^{(4)} \| M_b \| u_w).$$

The value of X is set to be k_t if $b = 0$, and otherwise k_w is randomly chosen from $\mathbb{Z}_q$. Let $\epsilon_5 = \Pr[b' = b]$ in this game. It is clear that this game is identical to Game_4, therefore $\epsilon_5 = \epsilon_4$ holds. Let $C_0 = (C_t^*, C_w^*)$ when $b = 0$, and $C_1 = (C_t^*, C_w^*)$ when $b = 1$. Distinguishing C_0 and C_1 is equivalent to distinguishing the following tuples.

$$(g_1^{y_t}, g_1^{v_t}, g_1^{y_t v_t k_t}, g_2^{h_t}, g_2^{h_t y_t}, g_1^{y_w}, g_1^{v_w}, g_1^{y_w v_w k_t}, g_2^{h_w}, g_2^{h_w y_w})$$
$$(g_1^{y_t}, g_1^{v_t}, g_1^{y_t v_t k_t}, g_2^{h_t}, g_2^{h_t y_t}, g_1^{y_w}, g_1^{v_w}, g_1^{y_w v_w k_w}, g_2^{h_w}, g_2^{h_w y_w})$$

It is straightforward to prove that to distinguish the above tuples is equivalent to distinguishing the extended DBDH tuples. Therefore, similar to proving semantic security of ElGamal scheme [16], it is straightforward to verify that $\epsilon_5 - \frac{1}{2}$ is negligible based on the extended DBDH assumption.

From the above analysis, we have that $|\epsilon_0 - \epsilon_5| \leq \Pr[Ent_1] + \Pr[Ent_2]$, which is negligible in the random oracle model based on the CDH assumption in $\mathbb{G}$ and the extended DBDH assumption. Note that $\epsilon = |\epsilon_0 - \frac{1}{2}|$ and $|\epsilon_5 - \frac{1}{2}|$ is negligible, then ϵ is negligible. The theorem now follows. □

Theorem 3. *The proposed FG-PKEET cryptosystem is IND-CCA secure against a Type-II adversary in the random oracle model based on the CDH assumption in $\mathbb{G}$ and the DDH assumption in $\mathbb{G}_1$.*

Proof sketch of Theorem 3. Suppose that an adversary has the advantage ϵ in the attack game shown in Fig. 4. The security proof is done through a sequence of games [16].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from the adversary, and all hash functions are treated as random oracles. Let $\epsilon_0 = \Pr[b' = b]$. Clearly, $\epsilon_0 = \epsilon$ holds.

Game_1: In this game, the challenger performs identically to that in Game_0 except that the following. For any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger computes $M_i \| u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)}$ and verifies $g^{u_i} = C_i^{(1)}$. If the verification fails, return $\perp$. Then, the challenger checks whether there exists an input query $C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i)$ to H_1, which outputs $C_i^{(5)}$. If such an input query exists, return M_i; otherwise return $\perp$. Let the event Ent_1 be that, for some C_i, a fresh input $C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i$ to H_1 results in $C_i^{(5)}$. Clearly, This game is identical to Game_0 unless the event Ent_1 occurs. It is straightforward that $\Pr[Ent_1]$ is negligible if H_1 is modeled as a random oracle. Let $\epsilon_1 = \Pr[b' = b]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_1 - \epsilon_0| \leq \Pr[Ent_1]$.

Game_2: In this game, the challenger performs identically to that in Game_1 except that, for any index i, if the adversary queries the decryption oracle Dec with C_i, the challenger does the following. Try to obtain the query to the oracle H_1 with the input $C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i$ satisfying

$$M_i||u_i = \mathsf{H}_2(g^{u_i x_i}) \oplus C_i^{(3)},\ g^{u_i} = C_i^{(1)},\ \mathsf{H}_1(C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i) = C_i^{(5)}.$$

If such a query cannot be found, return $\perp$. Otherwise, return M_i. This game is indeed identical to Game_1. Let $\epsilon_2 = \Pr[b' = b]$, then we have $\epsilon_2 = \epsilon_1$.

Game_3: In this game, the challenger performs identically to that in Game_2 except that the challenge C_t^* is generated as follows.

$$C_t^{(1)} = g^{u_t},\ C_t^{(2)} = g_1^{v_t}, \delta \in_R \{0,1\}^{m+d_1},\ C_t^{(3)} = \delta,$$

$$C_t^{(4)} = g_1^{v_t y_t} \cdot \mathsf{H}_3(M_b),\ C_t^{(5)} = \mathsf{H}_1(C_t^{(1)}||C_t^{(2)}||C_t^{(3)}||C_t^{(4)}||M_b||u_t).$$

This game is identical to Game_2 unless the event Ent_2 occurs, namely $g^{u_t x_t}$ is queried to the random oracle H_2. Note that the private key x_t is never used to answer the adversary's queries. Therefore, $\Pr[Ent_2]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_3 = \Pr[b' = b]$ in this game. From the Difference Lemma in [16], we have $|\epsilon_3 - \epsilon_2| \leq \Pr[Ent_2]$.

Game_4: In this game, the challenger performs identically to that in Game_3 except that the challenge C_t^* is generated as follows.

$$C_t^{(1)} = g^{u_t},\ C_t^{(2)} = g_1^{v_t}, \delta \in_R \{0,1\}^{m+d_1},\ C_t^{(3)} = \delta,$$

$$C_t^{(4)} = g_1^{v_t y_t} \cdot \mathsf{H}_3(M_b),\ \gamma \in_R \{0,1\}^{\ell},\ C_t^{(5)} = \gamma.$$

This game is identical to Game_3 unless $C_t^{(1)}||C_t^{(2)}||C_t^{(3)}||C_t^{(4)}||M_b||u_t$ is queried to the random oracle H_1, referred to as the event Ent_3. Let $\epsilon_4 = \Pr[b' = b]$ in this game. Based on the CDH in $\mathbb{G}$, we have $|\epsilon_4 - \epsilon_3| \leq \Pr[Ent_3]$ is negligible.

Just the same as in proving the semantic security of ElGamal scheme [16], it is straightforward to verify that $\epsilon_4 - \frac{1}{2}$ is negligible based on the DDH assumption in $\mathbb{G}_1$. From the above analysis, we have that $|\epsilon_0 - \epsilon_4| \leq \Pr[Ent_1] + \Pr[Ent_2] + \Pr[Ent_3]$, which is negligible in the random oracle model based on the CDH assumption in $\mathbb{G}$ and the DDH assumption in $\mathbb{G}_1$. Note that $\epsilon = |\epsilon_0 - \frac{1}{2}|$ and $|\epsilon_4 - \frac{1}{2}|$ is negligible, then ϵ is negligible. The theorem now follows. □

4.3 Potential Vulnerability and Enhancement

Note that since a Type-I adversary has access to a token $T_{i,t}$, then given a ciphertext $\mathsf{Enc}(M, PK_t)$ it can test whether $M' = M$ holds for any M' by checking the following equality

$$\mathsf{Com}(\mathsf{Enc}(M', PK_i), \mathsf{Enc}(M, PK_t), T_{i,t}) = 1.$$

Therefore, in the extreme situation when the actual message space $\mathcal{M}$ is polynomial size or the min-entropy of the message distribution is much lower than the security parameter, for FG-PKEET, a Type-I adversary (or, semi-trusted proxies) is capable of mounting an offline message recovery attack by checking every $M' \in \mathcal{M}$.

This type of attack is unavoidable due to the desired plaintext equality test functionality, similar to the offline keyword guessing attack in the case of PEKS (or searchable encryption) [11,18]. However, compared with the formulation in [20], where any adversary can mount the attack, our formulation achieves a significant security improvement because a Type-II adversary is unable to mount the attack. Although an offline message recovery attack is theoretically unavoidable in the presence of a Type-I adversary, but, depending on the specific cryptosystem, certain countermeasure can be employed to mitigate such an attack. One possible countermeasure is shown as below.

As in the original cryptosystem proposed in Section 3, the enhanced cryptosystem requires the same global parameters, namely

$$(\ell, \mathbb{G}, g, p, \mathsf{H}_1, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, q, \mathsf{H}_2, \mathsf{H}_3).$$

In addition, $Q \cdot T$, a puzzle hardness parameter L (detailed below), and a hash function $\mathsf{UH} : \{0,1\}^* \rightarrow \mathbb{Z}^*_{Q \cdot T}$ are also published, where Q, T are two large primes. These additional parameters are required by the computational client puzzle scheme [14], which is employed because it is deterministic and immune to parallel attacks [19]. Note that the generation of $Q \cdot T$ could be bootstrapped by a party trusted by all users in the system, and threshold techniques (e.g. [9]) can be used to improve the security. Nevertheless, this trust assumption is not required for achieving the existing security properties.

The algorithm KeyGen is identical to that in the original scheme, while the algorithms Enc and Dec are redefined as follows.

– $\mathsf{Enc}(M, PK)$: This algorithm outputs a ciphertext $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, where

$$u \in_R \mathbb{Z}_p,\ C^{(1)} = g^u,\ C^{(3)} = \mathsf{H}_2(g^{ux}) \oplus M \| u,\ v \in_R \mathbb{Z}_q,\ C^{(2)} = g_1^v,$$

$$C^{(4)} = g_1^{vy} \cdot \mathsf{H}_3((\mathsf{UH}(M))^{2^L} \mod Q \cdot T)),\ C^{(5)} = \mathsf{H}_1(C^{(1)} \| C^{(2)} \| C^{(3)} \| C^{(4)} \| M \| u).$$

– $\mathsf{Dec}(C, SK)$: This algorithm first computes $M' \| u' = C^{(3)} \oplus \mathsf{H}_2((C^{(1)})^x)$, and then check the following
  1. $g^{u'} = C^{(1)}$,
  2. $\mathsf{H}_1(C^{(1)} \| C^{(2)} \| C^{(3)} \| C^{(4)} \| M' \| u') = C^{(5)}$.

  If all checks pass, output M', otherwise output an error message $\perp$.

Compared with the original encryption and decryption algorithms, the main difference is in computing $C^{(4)}$, where the encryptor needs to perform L multiplications in $\mathbb{Z}^*_{Q\cdot T}$ in order to compute $(\mathsf{UH}(M))^{2^L} \mod Q \cdot T$ to form $C^{(4)}$. Let every user U_i, for $i \geq 1$, adopt the above public key encryption scheme, and U_i's key pair be denoted as (PK_i, SK_i). The algorithms Aut is identical to that in the original cryptosystem, but the Com algorithm is defined as follows.

– $\mathsf{Com}(C_i, C_j, T_{i,j})$: This algorithm outputs 1 if $x_i = x_j$ or 0 otherwise, where

$$
\begin{aligned}
x_i &= \frac{\hat{e}(C_i^{(4)}, g_2^{r_{i,j}})}{\hat{e}(C_i^{(2)}, g_2^{y_i r_{i,j}})} & x_j &= \frac{\hat{e}(C_j^{(4)}, g_2^{r_{i,j}})}{\hat{e}(C_j^{(2)}, g_2^{y_j r_{i,j}})}\\
&= \frac{\hat{e}(g_1^{v_i y_i} \cdot \mathsf{H}_3((\mathsf{UH}(M_i))^{2^L} \mod Q \cdot T)), g_2^{r_{i,j}})}{\hat{e}(g_1^{v_i}, g_2^{y_i r_{i,j}})} & &= \frac{\hat{e}(g_1^{v_j y_j} \cdot \mathsf{H}_3((\mathsf{UH}(M_j))^{2^L} \mod Q \cdot T)), g_2^{r_{i,j}})}{\hat{e}(g_1^{v_j}, g_2^{y_j r_{i,j}})}\\
&= \hat{e}(\mathsf{H}_3((\mathsf{UH}(M_i))^{2^L} \mod Q \cdot T)), g_2)^{r_{i,j}} & &= \hat{e}(\mathsf{H}_3((\mathsf{UH}(M_j))^{2^L} \mod Q \cdot T)), g_2)^{r_{i,j}}
\end{aligned}
$$

As to this enhanced cryptosystem, the existing properties still hold, and their security proofs remain exactly the same. If a proxy is given U_t's ciphertext $\mathsf{Enc}(M, PK_t)$ and token $T_{i,t}$, then it can obtain $\mathsf{H}_3((\mathsf{UH}(M))^{2^L} \mod Q \cdot T)$. To test any M', the most efficient approach for the proxy is to compute $(\mathsf{UH}(M'))^{2^L} \mod Q \cdot T$ and perform a comparison based on its hash value. Since every test will cost L multiplications, then by setting an appropriate L the offline message recovery attack will be made computationally very expensive. Suppose that the size of the actual message space is not very small, this approach will deter the attack to some extent.

It is worth noting that, in this enhanced cryptosystem, the encryptor needs to perform L multiplications to mask the message in the encryption. This may be a computational bottleneck for some application scenarios. How to overcome this drawback while still mitigating the attack is an interesting future work.

5 Conclusion

In this paper, we have proposed a new formulation for PKEET, namely FG-PKEET. Compared with the formulation in [20], we have introduced a fine-grained authorization mechanism for users to specify who can perform equality test between their ciphertexts and successfully mitigate the possible drawbacks. We believe that the new formulation suits theoretical and practical security requirements better, and will be an important building block in designing privacy protection solutions supporting operations on encrypted data. Beyond this work, there are many interesting future research directions. One is to investigate the security implications when the user set and the proxy set overlap in the case of FG-PKEET. Our feeling is that in that case OW-CCA is the strongest security we can achieve. Another line of research is to investigate the practical countermeasures against offline message recovery attacks in the extreme situation, when the message space is polynomial size or the min-entropy of the message distribution is much lower than the security parameter.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.
2. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Technical Report Report 2005/417, IACR, 2005. http://eprint.iacr.org/2005/417.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998.
4. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.
5. M. Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, 1983.
6. A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, 2009.
7. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
8. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.
9. D. Boneh and M. K. Franklin. Efficient generation of shared rsa keys (extended abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 425–439, 1997.
10. X. Boyen. The uber-assumption family. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography — Pairing 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, 2008.
11. J. W. Byun, H. S. Rhee, H.Park, and D. H. Lee. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In W. Jonker and M. Petkovic, editors, *Secure Data Management, Third VLDB Workshop, SDM 2006*, volume 4165 of *LNCS*, pages 75–83. Springer, 2006.
12. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO 1984*, volume 196 of *LNCS*, pages 10–18, 1985.
13. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238, 1999.
14. R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology, 1996.
15. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology— EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

16. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.
17. D. F. Sittig. Personal health records on the internet: a snapshot of the pioneers at the end of the 20th century. *I. J. Medical Informatics*, 65(1):1–6, 2002.
18. Q. Tang and L. Chen. Public-key encryption with registered keyword search. In *Proceeding of Public Key Infrastructure, 5th European PKI Workshop: Theory and Practice (EuroPKI 2009)*, volume 6391 of *LNCS*, pages 163–178. Springer, 2009.
19. Q. Tang and A. Jeckmans. On non-parallelizable deterministic client puzzle scheme with batch verification modes. Technical Report TR-CTIT-10-02, CTIT, University of Twente, 2010. http://eprints.eemcs.utwente.nl/17107/.
20. G. Yang, C. Tan, Q. Huang, and D. S. Wong. Probabilistic public key encryption with equality test. In J. Pieprzyk, editor, *Topics in Cryptology — CT-RSA 2010*, volume 5985 of *LNCS*, pages 119–131. Springer, 2010.