Public Key Encryption Schemes Supporting Equality Test with Authorization of Different Granularity

Qiang Tang

DIES, Faculty of EEMCS
University of Twente, the Netherlands
Drienerlolaan 5, 7522 NB Enschede, The Netherlands
E-mail: tonyrhul@gmail.com

Abstract: In this paper, we extended the work about public key encryption schemes supporting fine-grained authorization (FG-PKEET), done by Tang (2011b). First of all, we correct some flaws in Tang (2011b) and discuss how to extend the proposed cryptosystem to support approximate equality test. Secondly, we present a comparison between FG-PKEET and other similar primitives including AoN-PKEET by Tang (2011a) and PKEET by Yang et al. (2010), and demonstrate their differences in complexity and achieved security. Thirdly, to mitigate the inherent offline message recovery attacks, we extend FG-PKEET to a two-proxy setting, where two proxies need to collaborate in order to perform an equality test. Finally, we propose a cryptosystem and prove its security in the two-proxy setting.

Keywords: public key encryption; equality test; authorization granularity.

Biographical notes: Qiang Tang received his PhD degree from Royal Holloway, University of London in 2007. Between October 2006 and September 2007, he worked as a Podtdoc researcher at Ecole Normale Superieure, Paris, France. Currently, he is a research fellow at University of Twente, the Netherlands.

1 Introduction

Data and computation oursourcing is becoming a popular trend due to the potential economic benefits. 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. Recently, 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, which is generally referred to as PKEET.

As mentioned by Yang et al. (2010), PKEET is a useful building block in constructing secure solutions for outsourced databases. Besides, we can foresee more applications in the emerging computing scenarios. For example, Tang (2011a) shows that a special variant of PKEET cryptosystem (i.e. AoN-PKEET) can allow patients to encrypt their attributes and a semi-trusted proxy to match the encrypted attributes and recommend the patients to each other in an Internet-based PHR application by Sittig (2002).

1.1 Related Work

The concept of PKEET cryptosystem was proposed by Yang et al. (2010), and their formulation lacks an authorization mechanism for users to specify who can perform equality test between their ciphertexts. In reality, any entity can perform the equality test. As a result, standard semantic security or IND-CPA security cannot be achieved. A serious consequence is that, 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) by Byun et al. (2006); Tang and Chen (2009).

Tang (2011a) extends the concept of PKEET, and introduces an authorization mechanism for users to specify who can perform a plaintext equality test from their ciphertexts. The new primitive is denoted as AoN-PKEET. With an AoN-PKEET cryptosystem, every user can independently run an authorization algorithm to issue his token to some semi-trusted proxies. If a proxy receives the tokens from both Alice and Bob, then it is able to perform a plaintext equality test from their ciphertexts; otherwise, it cannot do so.

Tang (2011b) integrates a fine-grained authorization policy enforcement mechanism into PKEET and proposes 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 than with PKEET and AoN-PKEET.

- 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.

The concepts of PKEET has a close nature to that of Public key encryption with keyword search (PEKS) Boneh et al. (2004b) and public key encryption with registered keyword search (PERKS) Tang and Chen (2009). With a PEKS or PERKS scheme, a user can enable a server to perform equality test between the keyword embedded in a trapdoor and a tag (attached to a ciphertext), and the user enforces her authorization by issuing a trapdoor to the server. In particular, Hwang and Lee (2007) extend PEKS to a multi-user setting, where the tag contains keywords encrypted under a group of users' public keys. Compared with these primitives, PKEET in general (namely PKEET, AoN-PKEET, and FG-PKEET) is different in the sense that it aims at equality test between the plaintexts of any number of ciphertexts, while the PEKS and related primitives aim at testing the equality of keyword(s) in a given trapdoor and multiple tags.

1.2 Our Contribution

This paper is an extended work based on that by Tang (2011b). The contributions lie in four aspects. First of all, we correct some flaws in the description of proposed cryptosystem and its security proofs by Tang (2011b). In addition, we discuss how to extend the proposed cryptosystem to support approximate equality test based on the Euclidean distance metric. Secondly, we present a comparison between FG-PKEET and other similar primitives including AoN-PKEET by Tang (2011a) and by PKEET Yang et al. (2010), and demonstrate their differences in complexity and achieved security. Thirdly, to mitigate the inherent offline message recovery attacks, we extend FG-PKEET to a two-proxy setting, where two proxies need to collaborate in order to perform an equality test. we present a security model to formalize a set of security properties which are similar to those of FG-PKEET. Finally, we propose a cryptosystem based on the FG-PKEET cryptosystem proposed by Tang (2011b), and prove its security in our security model.

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 and prove its security. In Section 4, we describe two relevant properties for FG-PKEET cryptosystems, namely resistance to offline message recovery attacks and approximate equality test support. In Section 5, we compare FG-PKEET with PKEET and AoN-PKEET. In Section 6, we extend FG-PKEET to two-proxy setting, propose a cryptosystem and prove its security. In Section In Section 7, 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 2.1: 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.

Remark 2.1: In the definitions of Aut and Com, we implicitly assume that $i \neq j$ because, for the moment, we are only interested in testing the equality of the ciphertexts of two different users. This implies that, for a secure FG-PKEET cryptosystem, with a token $T_{i,j}$, the proxy may be able to test the equality of two ciphertexts of U_i. For example, in the proposed FG-PKEET cryptosystem in Section 3, the token $T_{i,j}$ actually allows the proxy to perform equality test between the ciphetrexts of U_i. Arguably, this may be regarded as a potential vulnerability or be a violation of the expected fine-grained authorization capability. In Section 6, we will extend the concept of FG-PKEET into a two-proxy setting, where we will take this issue into account in more detail.

2.2 The Security Model

Before describing the security model for FG-PKEET, we first informally distinguish three common trust assumptions used in formulating security properties of cryptographic protocols.

- The first and strongest trust assumption is *fully trusted*. If Alice is fully trusted in a protocol, then she will faithfully follow the protocol specification and do nothing else.

- The second trust assumption is *semi-trusted* or *honest-but-curious*. If Alice is semi-trusted in a protocol, then she will faithfully follow the protocol specification and may try to deduce some private information from the transcripts of protocol execution. However, Alice will not act malicious in order to gain more benefits. For example, Alice will not try to collude with another party.

- The third and weakest trust assumption is *untrusted*. If Alice is untrusted in a protocol, then she is supposed to do everything in order to gain some private information.

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.

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 Figure 1.

1. Type-I adversary represents the semi-trusted proxies with which U_t has run the algorithm Aut with. Referring to Figure 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 Figure 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.

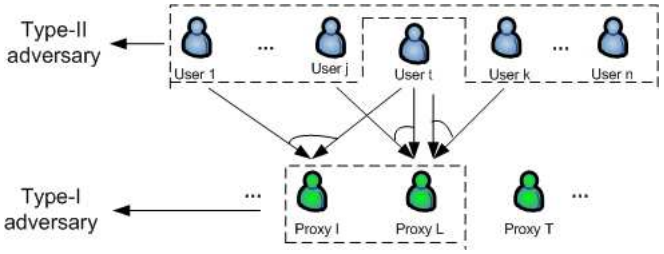

Figure 1 An Illustration of Adversaries for FG-PKEET

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. Referring to Figure 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 $\mathsf{Enc}(M, PK_k)$. 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 $C_t^* = \mathsf{Enc}(M_t, PK_t)$ 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 Figure 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 by Sahai and Waters (2005).

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.

2.2.1 OW-CCA Security against a Type-I Adversary

Definition 2.2: 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 Figure 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 $(i \neq j)$ 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'.

Figure 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 given by Bellare et al. (1998). One one hand, an IND-CPA secure scheme may not be OW-CCA. For instance, many homomorphic encryption schemes, such as the Elgamal scheme by ElGamal (1984) and the Paillier scheme by Paillier (1999), 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.

2.2.2 Fine-grained authorization property

Definition 2.3: 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 Figure 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 $(i \neq j)$ 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 $(t \neq w)$.
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'.

Figure 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.

2.2.3 IND-CCA Security against a Type-II Adversary

Definition 2.4: An FG-PKEET cryptosystem achieves IND-CCA security against a Type-II adversary, if, for any $1 \leq t \leq N$, any polynomial-time adversary has only a negligible advantage in the attack game shown in Figure 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 $1 \leq t \leq N$.
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, 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'.

Figure 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

3.1 Description of the 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. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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. $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{m+d_1}$, $\mathsf{H}_3 : \{0,1\}^* \to \mathbb{G}_1$ 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.

Remark 3.1: To satisfy the hardness assumptions required in Section 3.2, a Type-2 or Type-3 pairing setting as mentioned by Boyen (2008) is sufficient. A Type-2 setting means that there should be no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. A Type-3 setting means that there should be no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

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 $C^{(4)}$.

3.1.1 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 $1 \le t \le N$, 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 $SK_i = (x_i, y_i)$ and $PK_i = (g^{x_i}, g_1^{y_i})$, 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.

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

$$v_i \in_R \mathbb{Z}_q,\ C_i^{(2)} = g_1^{v_i},\ C_i^{(4)} = g_1^{v_i y_i} \cdot \mathsf{H}_3(M_i),$$

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

3.1.2 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 by Blum (1983). 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.

3.1.3 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

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

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

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. 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).

3.2 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.

3.2.1 Preliminary.

Following the work by Bellare and Rogaway (1993), we use random oracle to model hash functions in our security analysis. A function $P(k) : \mathbb{Z} \to \mathbb{R}$ 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, the DDH assumption in $\mathbb{G}_1$ is also referred to as the XDH (external Diffie-Hellman) assumption given by Boneh et al. (2004a).

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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, 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 $x_1, y_1, \in_R \mathbb{Z}_q$, 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-2 or Type-3 pairing setting mentioned by Boyen (2008). 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 given by Ballard et al. (2005), we can show the extended DBDH assumption is hard in the generic group model. We leave the details to the full paper.

3.2.2 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. Suppose an adversary has the advantage ϵ in the attack game shown in Figure 2. The security proof is done through a sequence of games by shoup (2006).

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 by shoup (2006), 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 $\epsilon_2 = \Pr[M_t' = M_t]$, 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 by shoup (2006), we have $|\epsilon_3 - \epsilon_2| \leq \Pr[Ent_2]$.

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 $\epsilon \leq \Pr[Ent_1] + \Pr[Ent_2] + \epsilon_3$, 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. Suppose an adversary has the advantage ϵ in the attack game shown in Figure 3. The security proof is done through a sequence of games by shoup (2006).

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 by shoup (2006), 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 by shoup (2006), we have $|\epsilon_3 - \epsilon_2| \leq \Pr[Ent_2]$.

$\mathsf{Game_4}$: In this game, the challenger performs identically to that as in $\mathsf{Game_3}$ 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 $\mathsf{Game_3}$, therefore $\epsilon_4 = \epsilon_3$ holds.

$\mathsf{Game_5}$: In this game, the challenger performs identically to that in $\mathsf{Game_4}$ 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 set to be k_w which 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 $\mathsf{Game_4}$, 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 by shoup (2006), 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. Suppose that an adversary has the advantage ϵ in the attack game shown in Figure 4. The security proof is done through a sequence of games by shoup (2006).

$\mathsf{Game_0}$: 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.

$\mathsf{Game_1}$: In this game, the challenger performs identically to that in $\mathsf{Game_0}$ 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 $\mathsf{Game_0}$ 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 by shoup (2006), we have $|\epsilon_1 - \epsilon_0| \leq \Pr[Ent_1]$.

$\mathsf{Game_2}$: In this game, the challenger performs identically to that in $\mathsf{Game_1}$ 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 $\mathsf{Game_1}$. Let $\epsilon_2 = \Pr[b' = b]$, then we have $\epsilon_2 = \epsilon_1$.

$\mathsf{Game_3}$: In this game, the challenger performs identically to that in $\mathsf{Game_2}$ 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 $\mathsf{Game_2}$ 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 by shoup (2006), we have $|\epsilon_3 - \epsilon_2| \leq \Pr[Ent_2]$.

$\mathsf{Game_4}$: In this game, the challenger performs identically to that in $\mathsf{Game_3}$ 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 $\mathsf{Game_3}$ 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 by shoup (2006), 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 Analysis beyond the Security Model

In this section, we first describe offline message recovery attacks, which apply to not only FG-PKEET but also PKEET and AoN-PKEET by Yang et al. (2010); Tang (2011a). Similar to the work by Tang (2011a), we show how to mitigate the attacks against the proposed FG-PKEET scheme in Section 3 by making use of computational puzzle schemes. Then, we present a variant of the proposed FG-PKEET cryptosystem to support approximate comparisons.

4.1 Offline Message Recovery Attack

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) by Byun et al. (2006); Tang and Chen (2009). However, compared with the formulation by Yang et al. (2010), 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 by Rivest et al. (1996), which is employed because it is deterministic and immune to parallel attacks by Tang and Jeckmans (2010). 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. Boneh and Franklin (1997)) can be used to improve the security. Nevertheless, this trust assumption is not required for achieving the existing security properties.

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

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

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

$$v \in_R \mathbb{Z}_q,\ C^{(2)} = g_1^v,$$

$$C^{(4)} = g_1^{vy} \cdot \mathsf{H}_3((\mathsf{UH}(M))^{2^L} \mod Q \cdot T)),$$

$$C^{(5)} = \mathsf{H}_1(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u).$$

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

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

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

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.

4.2 Approximate Equality Test Support

Note that, throughout the paper, we have only talked about exact equality test. It is reasonable to assume that, in some application scenarios, users may want the proxy to perform some form of approximate test. In this paper, we focus on the following case: if the plaintext messages are considered as integers, how to enable the proxy to test whether $|M_i - M_j| \leq T$, where T is an integer, given the ciphertexts for M_i and M_j. Note that approximate equality test based on other distance metrics may also be interesting, but we leave them for future work.

In the original FG-PKEET cryptosystem, the messages are hashed with H_3 so that there is no manipulation possible for the proxy to perform approximation test. We propose an variant cryptosystem, in which all algorithms remain the same except for Enc. Note that we treat messages as integers.

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

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

$$v \in_R \mathbb{Z}_q,\ C^{(2)} = g_1^v,\ C^{(4)} = g_1^{vy} \cdot g_1^M,$$

$$C^{(5)} = \mathsf{H}_1(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u).$$

It is clear that the only difference is the computation of $C^{(4)}$, which is defined to be $g_1^{vy} \cdot \mathsf{H}_3(M)$ in the original scheme.

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.

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

$$v_i \in_R \mathbb{Z}_q,\ C_i^{(2)} = g_1^{v_i},\ C_i^{(4)} = g_1^{v_i y_i} \cdot g_1^{M_i},$$

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

Recall from Section 3, the equality test algorithm of this variant cryptosystem is as follows.

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

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

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

Now, if the proxy wants to test whether $|M_i - M_j| \leq T$, then it can just test whether $x_i = x_j \cdot \hat{e}(g_1^t, g_2^{r_{i,j}})$ for any $-T \leq t \leq T$. With this variant, the proxy can perform other types of approximate equality tests, say testing whether $M_i = t \cdot M_j$ for any integer t.

It is true that the approximate equality test property could be very useful and necessary in some application scenarios. However, it should be noted that FG-PKEET cryptosystems with such a property may possess two vulnerabilities.

- The first one is offline message recovery attacks. Due to the requirement that the proxy somehow needs to be able to manipulate some operations in the comparison, the countermeasure proposed in Section 4.1 cannot be applied any more. It remains as a challenge to find a countermeasure.

- Besides recovering the messages, the proxy can figure out more information about the plaintexts, i.e. the relationships between the plaintexts. Note that this is directly resulted from approximate equality test requirement, and it demonstrates a conflict between the desired functionality and the available security.

5 Comparisons of PKEET Primitives

In this section, we compare the functionalities and achieved security of three similar primitives, including the PKEET by Yang et al. (2010), AoN-PKEET by Tang (2011a), and FG-PKEET proposed in this paper.

5.1 Review of PKEET

According to its definition, a PKEET cryptosystem (depicted in Figure 5) consists of four algorithms (KeyGen, Enc, Dec, Com), where (KeyGen, Enc, Dec) are similar to those of a standard public key encryption scheme and the Com algorithm allows any entity to compare two ciphertexts, which can be encrypted under a single user's public key or under two users' public keys respectively.

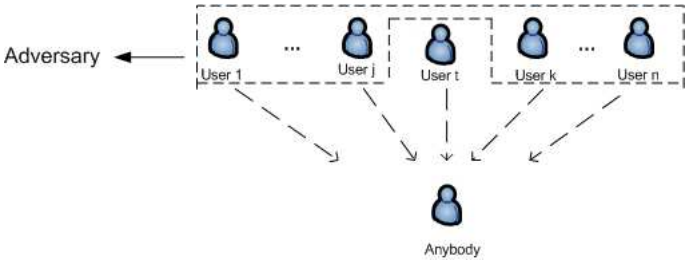

Figure 5 An Illustration of PKEET

As to security, for a user U_t, all other entities are potential adversaries, who have the same privilege in the sense that nobody has additional secret from U_t. As a result, a PKEET cryptosystem can only achieve OW-CCA security and it is naturally vulnerable to offline message recovery attacks, which can be mounted by any attacker. This means that a PKEET cryptosystem is weaker than an AoN-PKEET or an FG-PKEET cryptosystem from the security perspective.

As to the efficiency, a PKEET cryptosystem is clearly more efficient in the sense that the users do not need to explicitly authorize any proxy to enable the comparison, in contrast to the other two primitives. With respect to the common algorithms, generally a PKEET cryptosystem should also be more efficiently than an AoN-PKEET or FG-PKEET cryptosystem. With respect to the cryptosystems PKEET by Yang et al. (2010), AoN-PKEET by Tang (2011a), and FG-PKEET proposed in Section 3, a rough complexity comparison of the Enc, Dec, Com algorithms are shown in Table 1, where Exp means exponentiation.

	Enc	Dec	Com
PKEET	2 Exp	3 Exp	2 Pairings
AoN-PKEET	4 Exp	2 Exp	2 Exp
FG-PKEET	4 Exp	2 Exp	4 Pairings

Table 1 Computational Complexity Comparison

Note that the PKEET cryptosystem by Yang et al. (2010) uses Type 1 pairing, while the FG-PKEET cryptosystem proposed in Section 3 uses a Type 3 pairing mentioned by Boyen (2008).

5.2 Review of AoN-PKEET

An AoN-PKEET cryptosystem consists of the same set of algorithms as that of FG-PKEET, where (KeyGen, Enc, Dec) are identical to those of FG-PKEET. 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). The Aut and Com algorithms are defined as follows.

- $\mathsf{Aut}(SK_i)$: This algorithm takes the private key SK_i as input and outputs a token T_i.

- $\mathsf{Com}(C_i, C_j, T_i, T_j)$: This algorithm takes two ciphertexts C_i, C_j and two tokens T_i, T_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, and T_i, T_j are the tokens from U_i and U_j respectively. As a special case, if the proxy wants to perform equality test between U_i's ciphertexts, it only needs T_i to run Com.

With respect to an AoN-PKEET cryptosystem, for any honest user U_t, where $t \geq 1$, two types of adversaries are considered, as illustrated in Figure 6.

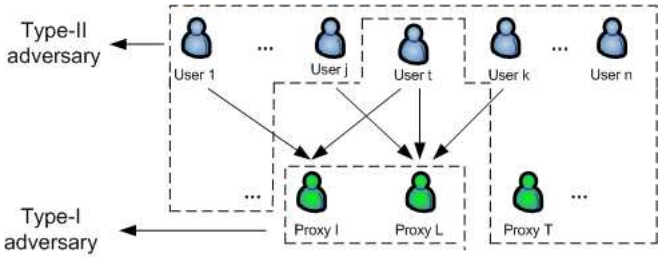

Figure 6 An Illustration of AoN-PKEET

1. Type-I adversary represents the semi-trusted proxies to which U_t has assigned his token. In addition, this type of adversary has access to the ciphertexts of all users. Referring to Figure 6, Proxy I and Proxy L are Type-I adversary. Against this type of adversary, OW-CCA security is defined, similar to that in Figure 2.

2. Type-II adversary represents all possibly malicious entities in the system from the perspective of U_t. In contrast to Type-I adversary, this type of adversary only has access to the ciphertexts of all users. Referring to Figure 6, such an adversary represents U_i $(i \geq 1, i \neq t)$, the untrusted proxies and any other outsider. Against this type of adversary, IND-CCA security is defined, similar to that in Figure 4.

In contrast to that of FG-PKEET, the Aut algorithm of AoN-PKEET only takes one private key as input, and the resulted token will enable the proxy to compare the user's ciphertexts to those of any other users. This implies that an AoN-PKEET cryptosystem has very coarse authorization capability, i.e. either giving full privilege or giving nothing to a proxy. As a result, the fine-grained authorization property defined in Figure 3 does not apply to AoN-PKEET.

6 FG-PKEET Cryptosystem in Two-Proxy Setting

Due to the nature of FG-PKEET, it is impossible to construct a cryptosystem which is immune to offline message recovery attacks and the caveat related to fine-grained authorization mentioned in Section 2.1. Therefore, we extend the concept of FG-PKEET into the two-proxy setting, where two semi-trusted proxies need to work together in order to perform a equality test. With respect to security, we assume that the semi-trusted proxies chosen by a user will not collude with each other. This is a standard assumption which has been used by many other threshold cryptographic primitives. For the simplicity of notation, we denote the new extended primitive as FG-PKEET$^+$.

6.1 New Security Model in Two-Proxy Setting

6.1.1 Description of FG-PKEET$^+$.

An FG-PKEET$^+$ cryptosystem consists of 5 algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aut}, \mathsf{Com})$, which are defined in the same way as for FG-PKEET, except for the Aut and Com algorithms.

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). Let all proxies be denoted as V_x $(1 \leq x \leq N')$, where N' is an integer. For simplicity of description, we require that there is no overlap between the user set and the proxy set. Suppose that U_i and U_j want to enable two proxies V_x and V_y to perform equality test between their ciphertexts, the Aut and Com algorithms are defined as follows.

- $\mathsf{Aut}(SK_i; SK_j; V_x, V_y)$: This algorithm is interactively run among U_i, U_j and two proxies V_x and V_y, and the two users use their private keys as their secret inputs while the proxies have no explicit input. At the end of the algorithm execution, proxy V_x receives a token $T_{x;i,j}$ as the output and proxy V_y receives a token $T_{y;i,j}$ as the output, while U_i and U_j receive no explicit output.

  We require that $i = j$ is allowed which means that a user want to authorize two proxies to perform equality test on his ciphertexts. But, $x \neq y$ should always hold which implies a two-proxy setting.

- $\mathsf{Com}(C_i, C_j, T_{x;i,j}; T_{y;i,j})$: This algorithm is interactively run between two proxies V_x, V_y, where C_i, C_j are the ciphertexts of U_i and U_j. At the end of the execution, the algorithm outputs 1 if $M_i = M_j$ or 0 otherwise for both proxies.

6.1.2 Correctness of FG-PKEET$^+$.

Similar to other cryptographic primitives, the basic requirement to 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 6.1: A FG-PKEET$^+$ cryptosystem achieves (unconditional) soundness if the following two equalities hold for any $1 \leq i, j \leq N$, $1 \leq x, y \leq N'$, and $M, M' \in \mathcal{M}$. Let $(PK_i, SK_i) = \mathsf{KeyGen}(\ell)$, $(PK_j, SK_j) = \mathsf{KeyGen}(\ell)$, and $(T_{x;i,j}, T_{y;i,j}) = \mathsf{Aut}(SK_i; SK_j; V_x, V_y)$.

1. $\mathsf{Dec}(\mathsf{Enc}(M, PK_i), SK_i) = M$, $\mathsf{Dec}(\mathsf{Enc}(M', PK_j), SK_j) = M'$.

2. $\mathsf{Com}(\mathsf{Enc}(M, PK_i), \mathsf{Enc}(M', PK_j), T_{x;i,j}, T_{y;i,j})$ outputs 1 if $M = M'$, and 0 otherwise to V_x and V_y.

6.1.3 Security Model of FG-PKEET$^+$.

For this new primitive, we make the same set of assumptions as we have made for FG-PKEET in Section 2. In addition, we assume that the semi-trusted proxies will not collude with each other. Note that this new assumption is only required for defining the fine-grained authorization property.

Similar to FG-PKEET, 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, as illustrated in Figure 7.

1. Type-I adversary represents any semi-trusted proxy with which U_t has run the algorithm Aut with. Referring to Figure 7, Proxy x, y, or z is Type-I adversary. The difference with the definition of Type-I adversary for FG-PKEET is that we assume there no collusion between semi-trusted proxies.

2. Type-II adversary represents all possibly malicious entities in the system from the perspective of U_t, namely U_i $(1 \leq i \leq N, i \neq t)$. This is identical to the definition of Type-II adversary for FG-PKEET.

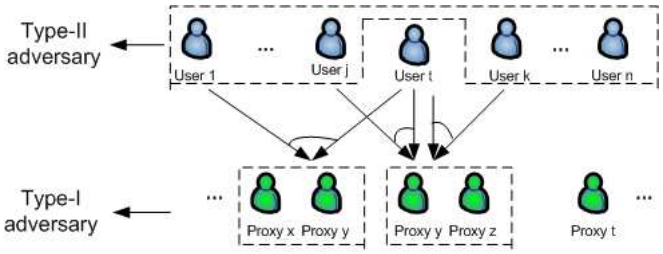

Figure 7 An Illustration of Adversaries for FG-PKEET$^+$

For the same reason as in the case of FG-PKEET, in the presence of a Type-I adversary, standard indistinguishability notions, such as IND-CCA and IND-CPA, cannot be achieved for FG-PKEET$^+$. Similarly, we focus on OW-CCA and fine-grained authorization properties, which are slightly different from those of FG-PKEET.

Definition 6.2: 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 Figure 8, where the advantage is defined to be $\Pr[M'_t = M_t]$.

Compared with the OW-CCA definition for FG-PKEET, the main difference is that $i = j$ is allowed in any Aut query. This reflects the fact that we take into account the fucntionality that a user needs to explicitly grant the authorization to proxies to perform equality test on his ciphertexts. Moreover, in the game, the adversary plays the role of both proxies in an Aut query. This implies that even if the proxies collude they cannot recover an encrypted message. It provides somewhat worst-case security guarantee.

Definition 6.3: 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 Figure 9, 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 $1 \leq i \leq N$.

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 user indexes i, j and proxy indexes x, y as input: the challenger runs the Aut algorithm with the adversary which plays the role of proxies x, y. Note that $i = j$ is allowed but $x = y$ is not allowed.

   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.

Figure 8 The Game for OW-CCA

Compared with the fine-grained authorization property definition for FG-PKEET, the main difference is that $i = j$ is allowed in any Aut query and $t = w$ is allowed in the challenge. This reflects the fact that we take into account the functionality that a user need to explicitly grant proxies the authorization to perform equality test on his ciphertexts, i.e. a proxy cannot perform equality test on two ciphertexts of U_i even if it has been authorized to perform equality test on two ciphertexts which are for U_i and U_j respectively.

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, which is identical to Definition 2.4, except for that an Aut oracle query involves two proxies which will be simulated by the challenger.

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 user indexes i, j and proxy indexes x, y as input: the challenger runs the Aut algorithm with the adversary which plays the role of proxy x. Note that $i = j$ is allowed but $x = y$ is not allowed.

   At some point, the adversary sends two integer indexes t, w to the challenger. 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'.

Figure 9 The Game for the Fine-grained Authorization Property

6.2 Description of the Proposed Cryptosystem

6.2.1 The Public Key Encryption Scheme.

The global parameters and the public key encryption algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ are defined as those of the FG-PKEET cryptosystem proposed in Section 3.1.

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 $SK_i = (x_i, y_i)$ and $PK_i = (g^{x_i}, g_1^{y_i})$, 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.

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

$$v_i \in_R \mathbb{Z}_q,\ C_i^{(2)} = g_1^{v_i},\ C_i^{(4)} = g_1^{v_i y_i} \cdot \mathsf{H}_3(M_i),$$

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

Suppose that all proxies are denoted as V_x $(1 \leq x \leq N')$, where N' is an integer.

6.2.2 The Token Generation Algorithm

Suppose that U_i and U_j want to authorize proxies V_x and V_y to perform equality test between their ciphertexts, then they run the following Aut algorithm interactively.

1. U_i and U_j generate $r_{i,j} \in_R \mathbb{Z}_q$ together. Note that, there can be many different ways for U_i and U_j to generate $r_{i,j}$. For instance, they can use an interactive coin flipping protocol, such as that of Blum (1983). Or, simply they can exchange two nonces and set $r_{i,j}$ to be the hash value of them.
2. U_i chooses $t_i \in_R \mathbb{Z}_q$, sends $(g_2^{r_{i,j}}, g_2^{(y_i - t_i) r_{i,j}})$ to proxy V_x, and sends $(g_2^{r_{i,j}}, g_2^{t_i r_{i,j}})$ to proxy V_y.
3. U_j chooses $t_j \in_R \mathbb{Z}_q$, sends $g_2^{t_j r_{i,j}}$ to proxy V_x, and sends $g_2^{(y_j - t_j) r_{i,j}}$ to proxy V_y.
4. Proxy V_x's token is $T_{x;i,j} = (g_2^{r_{i,j}}, g_2^{(y_i - t_i) r_{i,j}}, g_2^{t_j r_{i,j}})$, and proxy V_y's token is $T_{y;i,j} = (g_2^{r_{i,j}}, g_2^{t_i r_{i,j}}, g_2^{(y_j - t_j) r_{i,j}})$.

As a special case of the above algorithm, if $i = j$, then U_i first generates $r_{i,i}, t_i, t_i' \in_R \mathbb{Z}_q$ and then sends $T_{x;i,j}$ and $T_{y;i,j}$ to V_x and V_y respectively, where

$$T_{x;i,i} = (g_2^{r_{i,i}}, g_2^{(y_i - t_i) r_{i,i}}, g_2^{t_i' r_{i,i}}),$$

$$T_{y;i,i} = (g_2^{r_{i,i}}, g_2^{t_i r_{i,i}}, g_2^{(y_i - t_i') r_{i,i}}).$$

6.2.3 The Equality Test Algorithm.

Suppose that proxy V_x and proxy V_y have been authorized by U_i and U_j to perform equality test on their ciphertexts. If V_x wants to compare C_i and C_j, then the Com algorithm is interactively run between V_x and V_y as follows.

1. V_x sends the messages C_i, C_j to V_y, and they agree on the ciphertexts to perform equality test. If V_y has not been authorized by U_i and U_j, then it rejects V_x's request.
2. Proxy V_x computes X_i and sends $r_x \cdot X_i$ to proxy V_y, where $r_x \in_R \mathbb{G}_T$. Proxy V_y computes X_j and sends $r_y \cdot X_j$ to proxy V_x, where $r_y \in_R \mathbb{G}_T$.

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

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

3. Proxy V_x computes X_j^* and Proxy V_x computes X_i^*, where

$$X_j^* = r_x \cdot \frac{r_y \cdot X_j}{\hat{e}(C_j^{(2)}, g_2^{t_j r_{i,j}})}, \ X_i^* = r_y \cdot \frac{r_x \cdot X_i}{\hat{e}(C_i^{(2)}, g_2^{t_i r_{i,j}})}$$

4. Proxy V_x and Proxy V_x engages in a two-party protocol to compare X_i^* and X_j^*. At the end of the protocol, both proxies learn 1 if $X_i^* = X_j^*$ and 0 otherwise. As a specifical case of integer comparison problem, there are many solutions available, say that by Freedman et al. (2004).

As a special case of the above algorithm, if $i = j$ and V_x wants to compare C_i and C_i', then the Com algorithm is interactively run between V_x and V_y as follows. Let $C_i' = (C_i'^{(1)}, C_i'^{(2)}, C_i'^{(3)}, C_i'^{(4)}, C_i'^{(5)})$ be written in the following form.

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

$$v_i' \in_R \mathbb{Z}_q,\ C_i'^{(2)} = g_1^{v_i'},\ C_i'^{(4)} = g_1^{v_i' y_i} \cdot \mathsf{H}_3(M_i'),$$

$$C_i'^{(5)} = \mathsf{H}_1(C_i'^{(1)} || C_i'^{(2)} || C_i'^{(3)} || C_i'^{(4)} || M_i' || u_i').$$

1. V_x sends the messages C_i, C_i' to V_y, and they agree on the ciphertexts to perform equality test. If V_y has not been authorized by U_i, then it rejects V_x's request.

2. Proxy V_x computes X_i and sends $r_x \cdot X_i$ to proxy V_y, where $r_x \in_R \mathbb{G}_T$. Proxy V_y computes X_i' and sends $r_y \cdot X_i'$ to proxy V_x, where $r_y \in_R \mathbb{G}_T$.

3. Proxy V_x computes $X_i'^*$ and Proxy V_x computes X_i^*, where

$$X_i'^* = r_x \cdot \frac{r_y \cdot X_i'}{\hat{e}(C_i'^{(2)}, g_2^{t_i' r_{i,i}})}, \ X_i^* = r_y \cdot \frac{r_x \cdot X_i}{\hat{e}(C_i^{(2)}, g_2^{t_i r_{i,i}})}$$

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

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

4. Proxy V_x and Proxy V_x engage in a two-party protocol to compare $X_i'^*$ and X_i^*. At the end of the protocol, both proxies learn 1 if $X_i'^* = X_i^*$ and 0 otherwise.

From the description, it is straightforward to verify that the soundness property under Definition 6.1 is achieved.

6.3 Security Analysis

Note the fact that, for any i, j, the tokens $T_{x;i,j}$ and $T_{y;i,j}$ generated by the Aut algorithm proposed in Section 6.2.2 is a random division of the Token $T_{i,j}$ generated by the Aut algorithm proposed in Section 3.1.2. In other words, given $T_{i,j}$, we can faithfully generate $T_{x;i,j}$ and $T_{y;i,j}$. Therefore, with respect to the OW-CCA security property under Definition 6.2, a proof is identical to that of Theorem 1. Another fact about the Aut algorithm proposed in Section 6.2.2 is that U_i and U_j perform identically to the case in Section 3.1.2, regardless the trivial operation of dividing the values and assign them to two proxies. As a result, Therefore, with respect to the IND-CCA security property under Definition 2.4, a proof is identical to that of Theorem 3.

Next, we give a proof for the fine-grained authorization property according to Definition 6.3.

Theorem 4: *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. Suppose an adversary has the advantage ϵ in the attack game shown in Figure 9. Now, we consider two cases of an execution of the attack game. One case is that $t \neq w$ in the challenge. Based on the fact that, for any i, j, given $T_{i,j}$ we can faithfully generate $T_{x;i,j}$ and $T_{y;i,j}$. In this case, the adversary's privilege in the game is no more than that in the game defined in Figure 3. Therefore, based on Theorem 2, we have ϵ to be negligible.

Now, we consider the other case where $t = w$ in the attack game, and show that ϵ is also negligible. The security proof is done through a sequence of games by shoup (2006).

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 by shoup (2006), 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.

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

$$C_t^{(4)} = g_1^{v_t y_t} \cdot \mathsf{H}_3(M_0),\ \gamma_t \in_R \{0,1\}^{\ell},\ C_t^{(5)} = \gamma_t.$$

The challenge $C_t'^*$ is generated as follows.

$$C_t'^{(1)} = g^{u'_t},\ C_t'^{(2)} = g_1^{v'_t}, \delta'_t \in_R \{0,1\}^{m+d_1},\ C_t'^{(3)} = \delta'_t,$$

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

This game is identical to Game_3 unless the event Ent_3 occurs, namely a message containing u_t or u'_t is queried to the random oracle H_1. Therefore, $\Pr[Ent_2]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_4 = \Pr[b' = b]$ in this game. From the Difference Lemma by shoup (2006), we have $|\epsilon_4 - \epsilon_3| \leq \Pr[Ent_3]$.

Let $C_0 = (C_t^*, C_t'^*)$ when $b = 0$, and $C_1 = (C_t^*, C_t'^*)$ when $b = 1$. Distinguishing C_0 and C_1 is equivalent to distinguishing the following tuples:

$$(g_1^{y_t}, g_1^{v_t}, g_1^{y_t v_t} \cdot \mathsf{H}_3(M_0), g_1^{v'_t}, g_1^{y_t v'_t} \cdot \mathsf{H}_3(M_0))$$

$$(g_1^{y_t}, g_1^{v_t}, g_1^{y_t v_t} \cdot \mathsf{H}_3(M_0), g_1^{v'_t}, g_1^{y_t v'_t} \cdot \mathsf{H}_3(M_1)).$$

Note that every tuple can be regarded as two ElGamal encryptions. In particular, the adversary has two privileges:

- For some $1 \leq x \leq N'$, the adversary can possess $T_{x;i,j}$ for any $1 \leq i, j \leq N$ except for $i = j = t$, through issuing the Aut oracle queries. These tokens are random numbers without referring to the other parts, namely $T_{y;i,j}$.
- The adversary can issue Com oracle queries with two ciphertexts C_i, C_j and $T_{x;i,j}$ as input. Due to the randomization in step 2 and the secure two-party comparison protocol, this algorithm can be regarded as some sort of zero-knowledge "decryption and compare" oracle. This means that, given $C_i^{(2)} = g_1^{v_i}, C_i^{(4)} = g_1^{v_i y_i} \cdot h_i$ and $C_j^{(2)} = g_1^{v_j}, C_j^{(4)} = g_1^{v_j y_j} \cdot h_j$, the oracle returns 1 if $h_i = h_j$ and 0 otherwise without revealing any other information.

Given a tuple, these two privileges allow the adversary to exclude $\frac{T}{2}$ possibilities of whether the tuple contains two encryptions of $\mathsf{H}_3(M_0)$ or one encryption for $\mathsf{H}_3(M_0)$ and one encryption for $\mathsf{H}_3(M_1)$, given it can issue T

Com oracle queries. As a result, based on the DBDH assumption in the group $\mathbb{G}_1$, $\epsilon_4 - \frac{1}{2}$ is negligible in this game.

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 DBDH assumption. Note that $\epsilon = |\epsilon_0 - \frac{1}{2}|$ and $|\epsilon_4 - \frac{1}{2}|$ is negligible, then ϵ is negligible when $t = w$. Combining two cases, the theorem now follows. □

7 Conclusion

In this paper, we have reviewed the concepts of PKEET, AoN-PKEET, and FG-PKEET, and discussed their capabilities in authorizing users to control who can perform equality test on their ciphertexts and the available security guarantees. Our analysis has shown that offline message recovery attack is a security concern for all primitives, although only semi-trusted proxies can carry out the attack in the case of AoN-PKEET and FG-PKEET. To address the concern, we have proposed the concept of FG-PKEET$^+$, namely FG-PKEET in two-proxy setting. The tradeoff is clear: an FG-PKEET$^+$ cryptosystem can prevent offline message recovery attacks but it is more expensive to carry out the test because it requires an interactive protocol between two proxies. When to choose which primitive to use is depending on the security and efficiency requirements of the specific application scenario. It is an interesting future work to further investigate this. Recall from Section 6, one of the motivations of the two-proxy setting is to mitigate the caveat that the proxy can test equality of U_i's ciphertexts, given a token $T_{i,j}$. It remains as an interesting future work is to propose a FG-PKEET cryptosystem without this caveat, where the attack game for fine-grained authorization property is identical to that in Figure 3 except that $i = j$ and $t = w$ are allowed in the game.

References

Ballard, L., Green, M., de Medeiros, B., and Monrose, F. Correlation-resistant storage via keyword-searchable encryption. Tech. Rep. Report 2005/417, IACR, 2005. http://eprint.iacr.org/2005/417.

Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology — CRYPTO 1998* (1998), H. Krawczyk, Ed., vol. 1462 of *LNCS*, Springer, pp. 26–45.

Bellare, M., and Rogaway, P. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (1993), ACM Press, pp. 62–73.

Blum, M. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News 15*, 1 (1983), 23–27.

Boneh, D., Boyen, X., and Shacham, H. Short group signatures. In *Advances in Cryptology - CRYPTO 2004* (2004), M. K. Franklin, Ed., vol. 3152 of *LNCS*, Springer, pp. 41–55.

Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. Public Key Encryption with Keyword Search. In *Advances in Cryptology — EUROCRYPT 2004* (2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *LNCS*, Springer, pp. 506–522.

Boneh, D., and Franklin, M. K. Efficient generation of shared rsa keys (extended abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology* (1997), pp. 425–439.

Boyen, X. The uber-assumption family. In *Pairing-Based Cryptography — Pairing 2008* (2008), S. D. Galbraith and K. G. Paterson, Eds., vol. 5209 of *LNCS*, Springer, pp. 39–56.

Byun, J. W., Rhee, H. S., H.Park, and Lee, D. H. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In *Secure Data Management, Third VLDB Workshop, SDM 2006* (2006), W. Jonker and M. Petkovic, Eds., vol. 4165 of *LNCS*, Springer, pp. 75–83.

ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO 1984* (1985), G. R. Blakley and D. Chaum, Eds., vol. 196 of *LNCS*, pp. 10–18.

Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT 2004* (2004), vol. 3027 of *LNCS*, Springer, pp. 1–19.

Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT 1999* (1999), J. Stern, Ed., vol. 1592 of *LNCS*, pp. 223–238.

Rivest, R. L., Shamir, A., and Wagner, D. A. Time-lock puzzles and timed-release crypto. Tech. Rep. MIT/LCS/TR-684, Massachusetts Institute of Technology, 1996.

Sahai, A., and Waters, B. Fuzzy identity-based encryption. In *Advances in Cryptology—EUROCRYPT 2005* (2005), vol. 3494 of *LNCS*, Springer, pp. 457–473.

Shoup, V. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006. Accessed on November 10, 2011.

Sittig, D. F. Personal health records on the internet: a snapshot of the pioneers at the end of the 20th century. *I. J. Medical Informatics 65*, 1 (2002), 1–6.

Tang, Q. Public key encryption supporting plaintext equality test and user-specified authorization. Tech. Rep. TR-CTIT-11-20, CTIT, University of Twente, 2011. http://eprints.eemcs.utwente.nl/20529/.

Tang, Q. Towards public key encryption scheme supporting equality test with fine-grained authorization. In *Proceedings of the 16th Australasian Conference on Information Security and Privacy* (2011), vol. 6812 of *LNCS*, Springer, pp. 389–406.

Tang, Q., and Chen, L. Public-key encryption with registered keyword search. In *Proceeding of Public Key Infrastructure, 5th European PKI Workshop: Theory and Practice (EuroPKI 2009)* (2009), vol. 6391 of *LNCS*, Springer, pp. 163–178.

Tang, Q., and Jeckmans, A. On non-parallelizable deterministic client puzzle scheme with batch verification modes. Tech. Rep. TR-CTIT-10-02, CTIT, University of Twente, 2010. http://eprints.eemcs.utwente.nl/17107/.

Yang, G., Tan, C., Huang, Q., and Wong, D. S. Probabilistic public key encryption with equality test. In *Topics in Cryptology — CT-RSA 2010* (2010), J. Pieprzyk, Ed., vol. 5985 of *LNCS*, Springer, pp. 119–131.

Hwang, Y. H., and Lee, P. J. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *Pairing-Based Cryptography - Pairing 2007* (2007), T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, Eds., vol. 4575 of *LNCS*, Springer, pp. 2–22.