Public Key Encryption Supporting Plaintext Equality Test and User-Specified Authorization

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

August 1, 2011

Abstract

In this paper we investigate a category of public key encryption schemes which supports plaintext equality test and user-specified authorization. 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 plaintext equality test from their ciphertexts. We provide a formal formulation for this primitive, and present a construction with provable security in our security model. To mitigate the risks against the semi-trusted proxies, we enhance the proposed cryptosystem by integrating the concept of computational client puzzles. As a showcase, we construct a secure personal health record application based on this primitive.

1 Introduction

With the rapid advances in information technology, especially cloud computing, organizations and individuals have begun to transfer their data storage and processing from within their own organizational perimeters to those of third-party service providers. For example, quite a number of Internet-based personal health record (PHR) systems[23], such as Google Health [19] and Microsoft HealthVault [20], have been proposed so far. As a result of the transit, the outsourcers can successfully reduce their operating costs while providing better services. However, the downside is that, as many security critics have already pointed out, e.g. in [22], there are potential privacy risks for the outsourced data.

To tackle the privacy concerns, researchers have intensively worked on cryptographic encryption techniques that support operations on encrypted data. In this paper, we are interested in a category of public key encryption schemes, which supports plaintext equality test from ciphertexts generated under different public keys. We use the term PKEET (i.e. "Public Key Encryption supporting plaintext Equality Test") to refer to this category of public key encryption, and an informal description is given below.

> *Given a public key encryption scheme* (KeyGen, Enc, 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 any entity can 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'*.*

Previous Result. The concept of PKEET was proposed by Yang *et al.* [18]. Their formulation allows any entity to perform equality test, hence it lacks an authorization mechanism for users to specify who can perform a plaintext equality test. Inherently, it leads to the following security vulnerability. Given a ciphertext $\mathsf{Enc}(M, PK)$, any entity can determine whether $M = M'$ for any M' in the message space (note that this is implied by the equality test functionality). Considering the fact that ciphertexts are public information, it means that, given a ciphertext, any entity can discover a considerable amount of information about the plaintext. In many potential application scenarios, such as the PHR systems described in Section 5, the users may only want the privileged (or, semi-trusted) parties such as a hospital to perform the equality test, so that other un-privileged parties can learn nothing about their plaintexts. Clearly, a secure PKEET cryptosystem under the formulation in [18] may not suffice in these application scenarios.

Recently, Tang proposed the concept of PKEET with a fine-grained authorization mechanism [25]. In this primitive, two users must come together to generate a token which allows a proxy to compare their ciphertexts. While offering users with tight control over who can compare their ciphertexts, this may be a burden in some situations. Suppose that a user wants a proxy to compare his cipheretxts with those of his n classmates, then the user needs to generate a token with each of his classmates. And, the proxy needs to store n tokens for ciphertext comparison. In this case, it makes sense to have a PKEET with a simple or coarse-grained authorization mechanism. This motivates the new primitive AoN-PKEET, which will be proposed in this paper.

Contribution. We propose a new primitive AoN-PKEET, i.e. all-or-nothing PKEET, which introduces an authorization mechanism for users to specify who can perform a plaintext equality test from their ciphertexts. With a AoN-PKEET cryptosystem, every user can independently run the authorization algorithm (the Aut algorithm in our formulation in Section 2) 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. We use the term *all-or-nothing* because once a user, say Alice, has issued her token to a proxy, then this proxy is potentially able to perform equality test between Alice's ciphertexts and the ciphertexts of any other user, say Bob. The only prerequisite is that the proxy also obtains Bob's token, yet whether Bob will issue his token to the proxy is out of the control of Alice.

In the threat model of AoN-PKEET, we consider two types of adversaries whose main goal is to reveal information about the encrypted data (i.e. violating the confidentiality).

1. Type-I adversary represents semi-trusted proxies, who have access to both ciphertexts and users' tokens. With respect to such an adversary, we define the notion of OW-CCA security, namely one-wayness under a chosen ciphertext attack.

2. Type-II adversary represents all malicious entities, who only have access to users' ciphertexts. With respect to such an adversary, we define the notion of IND-CCA, namely indistinguishability under a chosen ciphertext attack.

We propose an AoN-PKEET cryptosystem, which achieves the OW-CCA security against a Type-I adversary and the IND-CCA security against a Type-II adversary. Both security properties are proven based on the Computational Diffie-Hellman (CDH) assumption in the random oracle model. To mitigate the potential risks against a Type-I adversary, we enhance the proposed cryptosystem by integrating the concept of computational client puzzles [14].

Based on AoN-PKEET, we construct a secure PHR application, in which patients can encrypt their PHRs and outsource the ciphertexts to a third-party Service Provider (SP). The SP is only required to be semi-trusted and cannot recover the plaintext PHRs, but it can still recommend patients to each other based on the encrypted PHRs. We note that a secure PKEET cryptosystem formulated in [18] cannot achieve the same level of security. Similarly, AoN-PKEET can also be used in building secure outsourced

database applications, which have been outlined by Yang *et al.* for PKEET [18]. In such applications, AoN-PKEET will also provide a higher level of security guarantees than PKEET.

Organization. The rest of the paper is organized as follows. In Section 2, we formulate the concept of AoN-PKEET. In Section 3, we provide a construction of AoN-PKEET and prove its security. In Section 4, we provide an enhanced AoN-PKEET cryptosystem based on computational client puzzles. In Section 5, we construct a secure personal health record (PHR) application based on AoN-PKEET. In Section 6, we briefly review the related work. In Section 7, we conclude the paper.

2 Formulation of AoN-PKEET

An AoN-PKEET cryptosystem consists of the algorithms (KeyGen, Enc, Dec), which are similar to those of standard public key encryption (PKE), as well as two new algorithms (Aut, Com). The algorithms are defined as follows.

- KeyGen(ℓ): 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.

- Enc(M, PK): This algorithm takes a message $M \in \mathcal{M}$ and the public key PK as input, and outputs a ciphertext C.

- 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 ($i \geq 1$), who adopt the above public key encryption scheme. For any i, suppose that U_i's key pair is denoted as (PK_i, SK_i). It is required that all users use the same message space $\mathcal{M}$. The Aut and Com algorithms are defined as follows.

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

- 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.

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.

Similar to the study of PKE schemes [2], an AoN-PKEET cryptosystem should be sound (or, correct). Informally, this property means that the algorithms Dec and Com work properly with valid inputs. Formally, it is defined as follows.

Definition 1 *An AoN-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. $\mathsf{Com}(\mathsf{Enc}(M, PK_i), \mathsf{Enc}(M', PK_j), \mathsf{Aut}(SK_i), \mathsf{Aut}(SK_j))$ *is equal to 1 if $M = M'$, and 0 otherwise.*

Besides the soundness property, we will define the security model and security properties for an AoN-PKEET cryptosystem in the following subsections.

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 The Threat Model

To facilitate our formal discussions, we make the following assumptions. All users honestly generate their public/private key pairs. Every token is sent to the proxy securely without being eavesdropped on by an outsider. Every proxy can serve multiple users to perform equality test. For any honest user U_t, where $t \geq 1$, he semi-trusts the proxies chosen by himself in the following sense.

1. The proxies will faithfully follow the protocol specifications in performing plaintext equality tests.

2. When considering message security, the proxies may behave maliciously. For example, in order to gain more information about $\mathsf{Enc}(M, PK_t)$, the proxies may send a string R to U_t to get the decryption result (this corresponds to the query to the decryption oracle in the security formulation shown in Figure 2).

3. Clearly, the proxies have more power than any other third-party adversary since they have access to U_t's token. We assume the proxies will not collude with others to disclose U_t's information. This is a natural assumption because in practice a user may trust one entity more than another one. As a result, it leads us to consider two types of adversaries in our formulation, as detailed below.

With respect to an AoN-PKEET cryptosystem, for any honest user U_t, where $t \geq 1$, we consider two types of adversaries, namely Type-I and Type-II adversaries as illustrated in Figure 1.

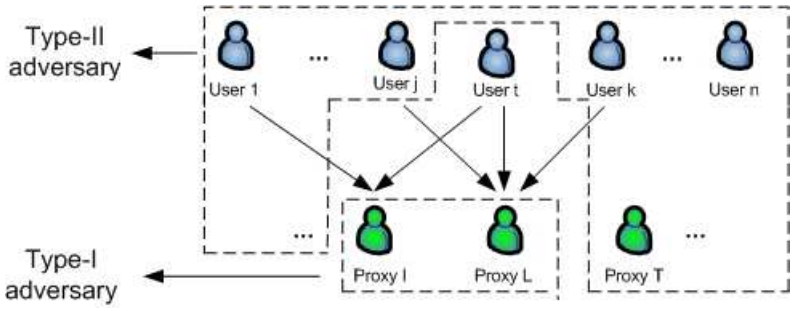

Figure 1: An Illustration of AoN-PKEET

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 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. In contrast to Type-I adversary, this type of adversary only has access to the ciphertexts of all users. Referring to Figure 1, such an adversary represents U_i ($i \geq 1, i \neq t$), the untrusted proxies and any other outsider.

2.2 OW-CCA Security against a Type-I Adversary

As to the power of a Type-I adversary, it receives U_t's token and may also obtain some information about U_t's plaintexts (i.e., has access to U_t's decryption oracle). In the context of AoN-PKEET, in the presence of a Type-I adversary, standard indistinguishability notions, such as IND-CCA and IND-CPA [2], cannot be achieved because of the desired plaintext equality test functionality. Therefore, we define the notion of one-wayness under a chosen ciphertext attack (OW-CCA).

As shown in Figure 2, the definition follows the conventional way, namely through an attack game between an adversary and a challenger which simulates the activities of the honest user U_t.

1. The challenger runs KeyGen to generate a public/private key pair (PK_t, SK_t).

2. Phase 1: The adversary is allowed to issue the following types of oracle queries.

   (a) Dec query with data C as input: the challenger returns $\mathsf{Dec}(C, SK_t)$.

   (b) Aut query: the challenger returns $\mathsf{Aut}(SK_t)$.

   At some point, the adversary asks the challenger for a challenge.

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^*. At some point, the adversary terminates by outputting a guess M_t'.

Figure 2: The Game for OW-CCA

Definition 2 *An AoN-PKEET cryptosystem achieves OW-CCA security against a Type-I adversary, if, for any* $t \geq 1$*, any polynomial-time adversary has only a*

negligible advantage in the attack game shown in Figure 2, where the advantage is defined to be $\Pr[M'_t = M_t]$.

2.3 IND-CCA Security against a Type-II Adversary

In the presence of a Type-II adversary, we define the notion of standard IND-CCA security. This definition is essentially identical to a standard one for public key encryption schemes. For clarity, we rephrase it in Figure 3.

1. The challenger runs KeyGen to generate a public/private key pair (PK_t, SK_t).

2. Phase 1: The adversary is allowed to issue the following types of oracle queries.

   (a) Dec query with data C: the challenger returns $\mathsf{Dec}(C, SK_t)$.

   At some point, the adversary sends two messages M_0, M_1 of equal length from $\mathcal{M}$ to the challenger for a challenge.

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 following restriction: *The* Dec *oracle should not have been queried with the data* C^*_t. At some point, the adversary terminates by outputting a guess b'.

Figure 3: The Game for IND-CCA

Compared with the OW-CCA definition against a Type-I adversary, the main difference here is that a Type-II adversary is not allowed to query the Aut oracle, namely without any access to U_t's token.

Definition 3 *An AoN-PKEET cryptosystem achieves IND-CCA security against a Type-II adversary, if, for any* $t \geq 1$*, any polynomial-time adversary has only a negligible advantage in the attack game shown in Figure 3, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

We have defined the CCA security against Type-I and Type-II adversaries. In both cases, it is straightforward to define the CPA security by simply disallowing the adversary's access to the Dec oracle in the attack games. We omit the detailed definitions in this paper.

2.4 Offline Message Recovery Attack

Note that since a Type-I adversary has access to U_t's token T_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

$$\mathsf{Com}(\mathsf{Enc}(M, PK_t), \mathsf{Enc}(M', PK_t), T_t, T_t) = 1.$$

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 AoN-PKEET, a Type-I adversary (or, semi-trusted proxies) is capable of mounting an offline message recovery attack by checking every $M' \in \mathcal{M}'$. For example, let the message space be the names of all diseases we know in the medical domain (as shown in the PHR application in Section 5), then it falls into the extreme situation.

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) [8, 16]. However, compared with PKEET formulated in [18], where any entity can mount the attack, our formulation achieves a significant security improvement because a Type-II adversary is unable to mount the attack. We further note that 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. In Section 4, we propose a countermeasure based on computational client puzzles [14] to secure the AoN-PKEET cryptosystem proposed in the next section.

3 The Proposed AoN-PKEET Cryptosystem

Note that an AoN-PKEET cryptosystem needs to achieve two functionalities: one is to enable a user to decrypt the ciphertext generated under his public key, the other is to enable a proxy to perform equality test on two ciphertexts. An immediate attempt of design is to combine a standard public key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with a deterministic one-way

function, such as a hash function H. Let user U_i have two key pair (PK_i, SK_i) and (PK_i', SK_i'), set the ciphertext for a message M to be

$$C = \mathsf{Enc}(M, PK_i), \mathsf{Enc}(\mathsf{H}(M), PK_i').$$

To assign his token to a proxy, U_i sends SK_i' to the proxy. However, this attempt does not achieve OW-CCA against a Type-I adversary under Definition 2 even if the standard PKE scheme is IND-CCA secure, because an attacker can manipulate a ciphertext without being noticed by the decryptor. To obtain the message in C, the attack is simply to query

$$\mathsf{Enc}(M, PK_i), \mathsf{Enc}(\mathsf{H}(M'), PK_i'),$$

where M' is a message chosen by the attacker. Such an attack may be mitigated by asking the user to check whether $\mathsf{H}(M') = \mathsf{H}(M)$ in the decryption, but this will leak the information whether $M = M'$ which means that IND-CCA security still will not be achieved.

In this section, we propose an AoN-PKEET cryptosystem, which inherits the basic idea of the above attempt and is not only more efficient but also secure in our security model.

3.1 The Proposed AoN-PKEET Cryptosystem

Let $\mathbb{G}$ be a multiplicative group of prime order p, g be a generator of $\mathbb{G}$, ℓ be a security parameter, and $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$ be three cryptographic hash functions

$$\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{m+d}, \ \mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p, \ \mathsf{H}_3 : \{0,1\}^* \to \{0,1\}^{\ell},$$

where m is a polynomial in ℓ, $\{0,1\}^m$ is the message space, and d is the bit-length of p. The values of $(\ell, \mathbb{G}, g, p, m, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$ serve as the global parameters for the AoN-PKEET cryptosystem. In practice, these parameters could be publicly standardized.

The intuition behind our construction is that, when encrypting a message, the encryption algorithm encrypts both the message and a checksum of this message, where the checksum is computed with a one-way function from the message. When a user wants to assign her token, he discloses part of his private key to a proxy so that the latter can recover the checksum but not the message. In the following, we first define the algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{KeyGen}(\ell)$: This algorithm outputs a private key $SK = (x, y)$, where $x, y \in_R \mathbb{Z}_p$, and the corresponding public key $PK = (g^x, g^y)$.

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

$$u, v \in_R \mathbb{Z}_p,\ C^{(1)} = g^u,\ C^{(2)} = g^v,\ C^{(3)} = \mathsf{H}_1(g^{ux}) \oplus M||u,$$

$$C^{(4)} = g^{\mathsf{H}_2(g^{vy})+M},\ C^{(5)} = \mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u).$$

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

  1. $g^{u'} = C^{(1)}$,
  2. $\mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u') = C^{(5)}$.

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

Suppose that every user U_i, for $i \geq 1$, 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 $PK_i = (g^{x_i}, g^{y_i})$, 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 forms.

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

$$C_i^{(4)} = g^{\mathsf{H}_2(g^{v_i y_i})+M_i},\ C_i^{(5)} = \mathsf{H}_3(C_i^{(1)}||C_i^{(2)}||C_i^{(3)}||C_i^{(4)}||M_i||u_i).$$

Suppose that U_i wants a proxy to perform equality test on his ciphertexts, then he runs the following Aut algorithm to generate the token T_i for the proxy.

- $\mathsf{Aut}(SK_i)$: This algorithm returns a token $T_i = y_i$.

Suppose that a proxy has received the tokens T_i and T_j, then it can run the following Com algorithm to perform equality test on the ciphetexts C_i and C_j, which are encrypted under PK_i and PK_j respectively.

- $\mathsf{Com}(C_i, C_j, T_i, T_j)$: This algorithm outputs 1 if $C_i^{(4)} \cdot g^{-\mathsf{H}_2((C_i^{(2)})^{T_i})} = C_j^{(4)} \cdot g^{-\mathsf{H}_2((C_j^{(2)})^{T_j})}$ or 0 otherwise.

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

We now briefly compare the efficiency with that of the PKEET cryptosystem [18]. A ciphertext of the proposed AoN-PKEET cryptosystem contains 5 elements, which is larger than that of the PKEET cryptosystem in which case the ciphertext contains 3 elements. A detailed computational complexity comparison is shown in Table 1. Note that Exp denotes an exponentiation, which is much less cheaper than a Pairing operation.

	Encryption	Decryption	Equality Test
AoN-PKEET	4 Exp	2 Exp	2 Exp
PKEET	2 Exp	3 Exp	2 Pairing

Table 1: Computational Complexity Comparison

3.2 Security Analysis

Following the work by Bellare and Rogaway [3], 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 $k \geq N_f$. The security of the proposed AoN-PKEET cryptosystem relies on the CDH assumption, defined as follows.

Definition 4 *Let $\mathbb{G}$ be a multiplicative group of prime order p, g be a generator of $\mathbb{G}$, and ℓ be a security parameter. The CDH assumption holds if, given g^a and g^b where $a, b \in_R \mathbb{Z}_p$, any polynomial-time adversary can compute g^{ab} with a negligible probability with respect to ℓ.*

Next, we prove two theorems with respect to the OW-CCA and IND-CCA security properties of the proposed AoN-PKEET cryptosystem.

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

Proof sketch. Suppose that an adversary has the advantage ϵ the attack game shown in Figure 2. The security proof is done through a sequence of games [15].

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 it aborts the game if any of the random oracles returns the same output with two different inputs (referred to as the event Ent_1). Clearly, $\Pr[Ent_1]$ is negligible if the hash functions are modeled as random oracles. Let $\epsilon_1 = \Pr[M'_t = M_t]$. From the Difference Lemma in [15], 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 the following event Ent_2 occurs. If the adversary queries the decryption oracle Dec with $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, the challenger returns $\perp$ if there is not an input $C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u$ to H_3 such that the oracle returns $C^{(5)}$. Clearly, $\Pr[Ent_2]$ is negligible if the hash functions are modeled as random oracles. Let $\epsilon_2 = \Pr[M'_t = M_t]$ in this game. From the Difference Lemma in [15], we have $|\epsilon_2 - \epsilon_1| \leq \Pr[Ent_2]$.

Game_3: In this game, the challenger performs identically to that in Game_2 except that, if the adversary queries Dec with $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, the challenger does the following. Check the queries to the oracle H_3 to see whether whether there is an input $C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u'$ satisfying

$$\mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u') = C^{(5)},\ g^{u'} = C^{(1)},$$

$$M''||u'' = \mathsf{H}_1(g^{u'x_t}) \oplus C^{(3)},\ M'' = M'\ \ u'' = u'.$$

If so return M', otherwise return $\perp$. This game is in fact identical to Game_2. Let $\epsilon_3 = \Pr[M'_t = M_t]$ in this game, then $\epsilon_3 = \epsilon_2$.

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

$$C^{(1)}_t = g^{u_t},\ C^{(2)}_t = g^{v_t}, \delta \in_R \{0,1\}^{m+d},\ C^{(3)}_t = \delta,$$

$$C^{(4)}_t = g^{\mathsf{H}_2(g^{v_t y_t}) + M_t},\ C^{(5)}_t = \mathsf{H}_3(C^{(1)}_t||C^{(2)}_t||C^{(3)}_t||C^{(4)}_t||M_t||u_t).$$

This game is identical to Game_3 unless the event Ent_3 occurs, namely $g^{u_t x_t}$ is queried to the random oracle H_1. Note that the private key x_t is never used to answer the adversary's queries. Therefore, $\Pr[Ent_3]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_4 = \Pr[M'_t = M_t]$ in this game. The rationale is quite straightforward. Suppose the $\Pr[Ent_3]$ is non-negligible then the challenger can solve the CDH problem with the probability $\Pr[Ent_3]$: given a CDH challenge (g^a, g^b), the challenger sets g^{x_t} and g^{u_t} to be g^a and g^b respectively, and then randomly chooses an input to the random oracle H_1

as a guess for g^{ab}. From the Difference Lemma in [15], we have $|\epsilon_4 - \epsilon_3| \leq \Pr[Ent_3]$.

Since H_3 is modeled as a random oracle, it is clear that ϵ_4 is negligible if the discrete log computation is infeasible which is certainly true based on the CDH assumption. From the above analysis, we have that $\epsilon \leq \Pr[Ent_1] + \Pr[Ent_2] + \Pr[Ent_3] + \epsilon_4$, which is negligible based on the CDH assumption in the random oracle model. The theorem now follows. □

Theorem 2 *The proposed AoN-PKEET cryptosystem is IND-CCA secure against a Type-II adversary based on the CDH assumption in $\mathbb{G}$ in the random oracle model.*

Proof sketch Suppose an adversary has the advantage ϵ the attack game shown in Figure 3. The security proof is done through a sequence of games [15].

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, $|\epsilon_0 - \frac{1}{2}| = \epsilon$ holds.

Game_1: In this game, the challenger performs identically to that in Game_0 except that it aborts the game if any of the random oracles returns the same output with two different inputs (referred to as the event Ent_1). Clearly, $\Pr[Ent_1]$ is negligible if the hash functions are modeled as random oracles. Let $\epsilon_1 = \Pr[b' = b]$. From the Difference Lemma in [15], 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 the following event Ent_2 occurs. If the adversary queries the decryption oracle Dec with $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, the challenger returns $\perp$ if there is not an input $C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u$ to H_3 such that the oracle returns $C^{(5)}$. Clearly, $\Pr[Ent_2]$ is negligible if the hash functions are modeled as random oracles. Let $\epsilon_2 = \Pr[b' = b]$ in this game. From the Difference Lemma in [15], we have $|\epsilon_2 - \epsilon_1| \leq \Pr[Ent_2]$.

Game_3: In this game, the challenger performs identically to that in Game_2 except that, if the adversary queries Dec with $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$, the challenger does the following. Check the queries to the oracle H_3 to see whether there is an input $C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u'$ satisfying

$$\mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u') = C^{(5)},\ g^{u'} = C^{(1)},$$

$$M''||u'' = \mathsf{H}_1(g^{u'x_t}) \oplus C^{(3)},\ M'' = M'\ \ u'' = u'.$$

If so return M', otherwise return $\perp$. This game is in fact identical to Game_2. Let $\epsilon_3 = \Pr[b' = b]$ in this game, then $\epsilon_3 = \epsilon_2$.

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

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

$$\Gamma \in_R \mathbb{G},\ C_t^{(4)} = \Gamma,\ C_t^{(5)} = \mathsf{H}_3(C_t^{(1)} \| C_t^{(2)} \| C_t^{(3)} \| C_t^{(4)} \| M_t \| u_t).$$

This game is identical to Game_3 unless the event Ent_3 occurs, namely $g^{u_t x_t}$ is queried to the random oracle H_1 or $g^{v_t y_t}$ is queried to the random oracle H_2. Note that the private keys x_t, y_t are never used to answer the adversary's queries. Therefore, $\Pr[Ent_3]$ is negligible based on the CDH assumption in $\mathbb{G}$. Let $\epsilon_4 = \Pr[M_t' = M_t]$ in this game. The rationale is the same as that in the analysis of Game_4 in proving Theorem 1. From the Difference Lemma in [15], we have $|\epsilon_4 - \epsilon_3| \leq \Pr[Ent_3]$.

Since H_3 is model as a random oracle, it is clear that $|\epsilon_4 - \frac{1}{2}|$ is negligible. 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 $\mathbb{G}$. 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 An Enhanced AoN-PKEET Cryptosystem

In this section, we propose an enhanced AoN-PKEET cryptosystem based on the computational client puzzle scheme proposed in [14] to mitigate the offline message recovery attack, which is discussed in Section 2.4.

4.1 Offline Message Recovery Attack

In Section 2.4, we have shown a generic offline message recovery attack against any AoN-PKEET cryptosystem. Referring to the AoN-PKEET cryptosystem proposed in Section 3.1, from a ciphertext $\mathsf{Enc}(M, PK_t)$, a Type-I adversary with the token T_t can obtain g^M. As a result, a more efficient approach to mount the attack is to pre-compute $\{g^{M'} | M' \in \mathcal{M}'\}$, then the attack is simply a table lookup.

It is worth noting that only a Type-I adversary is capable of mounting an offline message recovery attack. Compared with PKEET formulated in [18], where any adversary can mount the attack, AoN-PKEET (and the proposed AoN-PKEET cryptosystem) achieves a significant security improvement.

4.2 The Enhanced AoN-PKEET Cryptosystem

As shown in the previous subsection, a Type-I adversary (or, a semi-trusted proxy) can mount the attack in a brute-force manner, namely try all the possible messages until finding a match. In theory, this type of attack is unavoidable due to the desired equality test functionality. Based on these facts, to mitigate the attack, a natural direction is to make the attack computationally expensive so that it will become computationally impossible for the adversary to mount the attack. To achieve the purpose, we make use of the computational client puzzle schemes [14, 24], which enable a prover to prove to a verifier that a certain amount of computing resources has been dedicated to solve a puzzle. The intuition is that the adversary is forced to solve a puzzle before being able to test a possible message.

In the enhanced cryptosystem, we choose the RSW scheme in [14], because it is proven secure and is deterministic and immune to parallel attacks [17]. These properties guarantee that a Type-I adversary cannot accelerate the attack by employing multiple computers to work in parallel. As in the original cryptosystem proposed in Section 3, the enhanced cryptosystem requires the same global parameters $(\ell, \mathbb{G}, g, p, m, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$. 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]. 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. [7]) can be used to improve the security. Nevertheless, this trust assumption is not required for achieving the OW-CCA and IND-CCA 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

$$u, v \in_R \mathbb{Z}_p,\ C^{(1)} = g^u,\ C^{(2)} = g^v,\ C^{(3)} = \mathsf{H}_1(g^{ux}) \oplus M \| u,$$

$$C^{(4)} = (\mathsf{H}_2(g^{vy}) + \mathsf{H}_2((\mathsf{UH}(M))^{2^L} \mod Q \cdot T)) \mod p,$$

$$C^{(5)} = \mathsf{H}_3(C^{(1)} \| C^{(2)} \| C^{(3)} \| C^{(4)} \| M \| u).$$

- $\mathsf{Dec}(C, SK)$: This algorithm first computes $M' \| u' = C^{(3)} \oplus \mathsf{H}_1((C^{(1)})^x)$, and then checks the following

1. $g^{u'} = C^{(1)}$,
2. $\mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M'||u') = C^{(5)}$.

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

Compared with the original encryption and decryption algorithms, the main difference lies in computing $C^{(4)}$, where the encryptor needs to perform L multiplications in computing $(\mathsf{UH}(M))^{2^L} \mod Q \cdot T$ 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.

- $\mathsf{Com}(C_i, C_j, T_i, T_j)$: This algorithm outputs 1 if $C_i^{(4)} - \mathsf{H}_2((C_i^{(2)})^{T_i}) \equiv C_j^{(4)} - \mathsf{H}_2((C_j^{(2)})^{T_j}) \pmod{p}$ or 0 otherwise.

Note that the enhancement does not incur any more work for the comparison algorithm compared with the original scheme.

As to this enhanced cryptosystem, the OW-CCA and IND-CCA properties still hold, and their security proofs are exactly the same as in Theorem 1 and Theorem 2. If a proxy is given U_t's ciphertext $\mathsf{Enc}(M, PK_t)$ and token T_t (i.e. y_t), then it can obtain $\mathsf{H}_2((\mathsf{UH}(M))^{2^L} \mod Q \cdot T)$. 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. 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 A Secure Internet-based PHR Application

In this section, we first present an overview of existing Internet-based PHR systems and point out the security risks. Then, based on AoN-PKEET, we propose a secure Internet-based PHR application, which allows patients to encrypt their data yet can still enjoy some sort of recommendation services.

5.1 Overview of Internet-based PHR Systems

A PHR is typically a collection of health data maintained by an individual, referred to as a patient. Recently, Internet-based PHR systems have received a lot of attention, some examples include Google Health [19] and Microsoft HealthVault [20]. Internet-based PHR systems typically help patients store their PHRs and allow the information to be accessed and edited via a web browser or some APIs, and they may also help patients find kindred spirits (i.e. build social networks) and share their information. Figure 4 shows a general picture of an Internet-based PHR system.

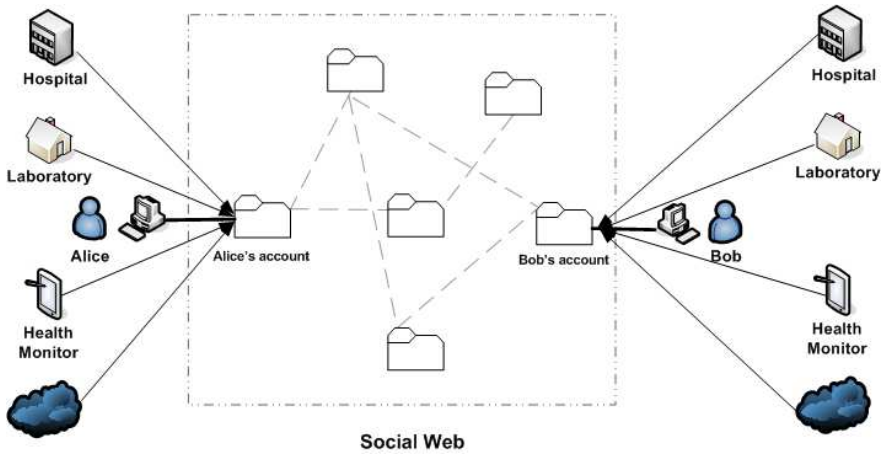

Figure 4: An Illustration of Outsourced PHRs

Considering a patient, say Alice, her PHR data can come from a lot of sources. From example, she can get prescription results from her doctor, treatments from a hospital, test results from a laboratory, and monitoring results from home-based sensors. In many systems, such as Google Health and Microsoft HealthVault, a lot of Alice's PHR data may be directly sent to Alice's account, while the rest will be input by Alice herself.

In most existing Internet-based PHR systems, patients will be provided privacy controls. However, there are a number of concerns which stop patients from sharing their data. One concern is that the system providers, say Microsoft or Google, are always able to fully access the data. Although there will be some privacy agreement, but patients may still worry about that these providers may abuse their data. The other concern is that, even if the service providers behave honestly, their databases may be compro-

mised, in which case all data may be leaked. Since PHRs are sensitive information to individuals, and an information leakage may cause undesirable consequences, such as being discriminated by the potential employer because of a disease.

5.2 The Proposed Application

With an AoN-PKEET cryptosystem, a PHR application can be built as follows. Alice generates a key pair (PK_a, SK_a) and publishes PK_a. When an entity, say a hospital or a laboratory, wants to contribute Alice's PHR data into the PHR system, it encrypts the data using PK_a and sends the ciphertext to Alice's account in the system. When Alice wants to match her PHR data with that of others, she can choose a semi-trusted entity as a proxy and assign her token. Later on, if other users also choose the same proxy then PHR data matching can be done through running the Com algorithm. The security of this application lies in the fact that patients' PHR data is always encrypted with their own public keys. Even if the storage of the PHR system is compromised, no information will be leaked. With respect to the semi-trusted proxies, in some extreme cases as discussed in Section 2.4, they may be able to obtain some side information about patients' PHR data from mounting an offline message recovery attack. But, when a cryptosystem such as that in Section 4 is used, the risk of offline message recovery attack can be reduced. It is worth noting that the above proposal serves as an example on how to use AoN-PKEET. In practice, it may be augmented with other security mechanisms to obtain a more secure PHR application.

As mentioned before, PHR data may be contributed by many different sources. We note that this may cause a problem in practice: data from different sources may have different forms or even been truncated. In this happens, the proposed solution of using AoN-PKEET to compare messages in an exact manner may not be desirable. To overcome this problem, we may have two methods, which can be employed in parallel in practice.

- One is to standardize data representations in the system, and make sure that all data contributors represent data in the same form.
- The other is to use an AoN-PKEET cryptosystem, which supports fuzzy comparison. In fact, the proposed AoN-PKEET scheme in Section 3 supports certain types of fuzzy comparison. Recall that the comparison algorithm is defined as follows:

– $\mathsf{Com}(C_i, C_j, T_i, T_j)$: This algorithm outputs 1 if $X = Y$ or 0 otherwise, where

$$X = C_i^{(4)} \cdot g^{-\mathsf{H}_2((C_i^{(2)})^{T_i})}, \; Y = C_j^{(4)} \cdot g^{-\mathsf{H}_2((C_j^{(2)})^{T_j})}.$$

Note that C_i is an encryption of M_i and C_j is an encryption of M_j.

In the above algorithm, we have $X = g^{M_i}$ and $Y = g^{M_j}$. Hence, for any integer h, the proxy can test whether $M_i = M_j + h$ by evaluating whether $X = Y \cdot g^h$. As a result, the proxy can perform certain types of fuzzy comparison in a brute-force manner. For example, if the messages are treated as integers, the proxy can test whether the Euclidean distance between M_i and M_j, namely $|M_i - M_j|$ is below a threshold. For the proposed solution, this kind of fuzzy comparison support may suffice.

We note that, with the proposed AoN-PKEET scheme, it may not be easy to perform other types of fuzzy comparison, such as based on the metric of the hamming distance between messages. We leave it as a future work to further investigate fuzzy comparisons with AoN-PKEET.

6 Other Related Work

In the literature, there have been enormous research efforts to investigate encryption techniques that support operations on encrypted data. Next, we provide a brief review on some related work to ours.

The concept of PKEET has a close nature to that of Public key encryption with keyword search (PEKS) [6] and public key encryption with registered keyword search (PERKS) [16]. 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.* [5]. 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.

The concept of PKEET also shares some similarity with that of proxy re-encryption (PRE) [4, 11]. With a PRE cryptosystem, a delegator Alice can issue a re-encryption key to a proxy so that the proxy can convert ciphertexts encrypted under Alice's public key into ciphertexts which can be decrypted by a delegatee Bob. During the re-encryption process, the proxy will learn nothing about the involved plaintexts. Yet, another related concept is homomorphic encryption, which is first proposed by Rivest, Adleman, and Dertouzos [13], and investigated by many others, e.g. ElGamal [9], Pailler [12], Gentry [10]. With a homomorphic encryption scheme, given two ciphertexts, any entity can compute a new ciphertext which is an encryption of the addition/multiplication/XOR/$\cdots$ of the plaintexts in the given ciphertext, depending on the homomorphic property. It worth noting that, for both PRE and homomorphic encryption, the proxy and any third party are not allowed to learn any information about the involved plaintexts. This property also differs from the purpose of PKEET, in which the proxy learns the equality status of the encrypted plaintexts.

7 Conclusion

In this paper, we have proposed a formulation and a construction for AoN-PKEET, namely public key encryption schemes which support plaintext equality test and user-specified authorization. Compared with PKEET formulated in [18], AoN-PKEET introduces a simple authorization mechanism for users to specify who can perform plaintext equality test from their ciphertexts. We believe that AoN-PKEET will be an important building block in designing privacy protection solutions (e.g. secure PHR applications) supporting operations on encrypted data. There are many interesting future works, including the following.

- In the security model for AoN-PKEET, we do not consider the consistency property of the encryption. Take the proposed scheme as an example, there is no way for the proxy to check whether $C = (C^{(1)}, C^{(2)}, C^{(3)}, C^{(4)}, C^{(5)})$ is a valid ciphertext or not, where

$$u, v \in_R \mathbb{Z}_p,\ C^{(1)} = g^u,\ C^{(2)} = g^v,\ C^{(3)} = \mathsf{H}_1(g^{ux}) \oplus M||u,$$

$$C^{(4)} = g^{\mathsf{H}_2(g^{vy})+M'},\ C^{(5)} = \mathsf{H}_3(C^{(1)}||C^{(2)}||C^{(3)}||C^{(4)}||M||u).$$

Clearly, given that $M \neq M'$, then C is not a valid ciphertext. We notice that this issue exists for the PKEET scheme proposed by Yang *et al.* [18] and the primitive in [25]. In addition, we notice that similar issue exists for the existing hybrid primitive of PEKS and PKE schemes, e.g. [26], where the encrypted keywords and messages may not be consistent with each other.

Though, the lack of this consistency will not affect the confidentiality of the data for AoN-PKEET, it may become a problem in some applications. For instance, a malicious message sender can generate an inconsistent encryption as shown above, then the proxy will generate wrong comparison results. We foresee two directions to solve this problem. One is to formalize a consistency property in the security model of AoN-PKEET and propose new schemes with such a property. The other one is, for the application which requires this property, to investigate auxiliary countermeasures to be used together with AoN-PKEET. For example, one possible countermeasure could be to record the connection between the message sender and the messages he has generated. If an inconsistent encryption is detected, then the message sender can be punished. We leave further investigation of the issue to be a future research work.

- In Section 5.2, we have noted that the proposed AoN-PKEET scheme may support certain types of fuzzy comparison but may not support other types of fuzzy comparison, such as that based on the metric of the hamming distance between messages. Here, we also note that fuzzy comparison cannot be achieved by the enhanced AoN-PKEET scheme described in Section 4 because the messages are hashed in $C_i^{(4)}$ and $C_j^{(4)}$. We leave it as a future work to further investigate AoN-PKEET schemes which support different types of fuzzy comparison. Moreover, for such schemes, it is an interesting research work is to investigate countermeasures against offline message recovery attacks against a Type-I adversary.

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 Press, 2004.

[2] 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.

[3] 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.

[4] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, 2009.

[6] 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.

[7] D. Boneh and M. K. Franklin. Efficient generation of shared rsa keys (extended abstract). In *Advances in Cryptology — CRYPTO 1997*, pages 425–439. Springer, 1997.

[8] 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.

[9] 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. Springer, 1985.

[10] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[11] M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A(1):54–63, 1997.

[12] 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. Springer, 1999.

[13] R. L. Rivest, L. M. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.

[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] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.

[16] 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.

[17] 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/.

[18] 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.

[19] Google Inc. Google Health. http://www.google.com/intl/nl/health/about/.

[20] Microsoft Corporation. Microsoft HealthVault. http://www.healthvault.com/personal/index.aspx.

[21] CureTogether. CureTogether. http://www.curetogether.com/.

[22] Cloud Security Alliance. Top Threat to Cloud Computing V1.0. http://www.cloudsecurityalliance.org/topthreats.html.

[23] 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.

[24] A. Juels and J. G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1999*, pages 151–165, 1999.

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

[26] R. Zhang and H. Imai. Generic Combination of Public Key Encryption with Keyword Search and Public Key Encryption. In F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, editors, *Cryptology and Network Security, 6th International Conference, CANS 2007*, volume 4856 of *LNCS*, pages 159–174. Springer, 2007.