Revisit the Concept of PEKS: Problems and a Possible Solution

Qiang Tang
Distributed and Embedded Systems Group
Faculty of EWI, University of Twente, the Netherlands
q.tang@utwente.nl

August 28, 2008

Abstract

Since Boneh *et al.* propose the concept, non-interactive Public-key Encryption with Keyword Search (PEKS) has attracted lots of attention from cryptographers. Non-interactive PEKS enables a third party to test whether or not a tag, generated by the message sender, and a trapdoor, generated by the receiver, contain the same keyword without revealing further information. In this paper we investigate a non-interactive PEKS application proposed by Boneh *et al.* and show our observations, especially that privacy is not protected against a curious server. We propose the notion of interactive PEKS, which, in contrast to non-interactive PEKS, requires the tag to be generated interactively by the message sender and the receiver. For this new primitive, we identify two types of adversaries, namely a curious user and a curious server, and provide security formulations for the desirable properties. We propose a construction for interactive PEKS and prove its security in the proposed security model.

1 Introduction

Non-interactive Public-key Encryption with Keyword Search (PEKS), proposed by Boneh *et al.* [6], allows message senders to reposit an encrypted message together with some tags (encrypted keywords) at a server, where the encryptions are based on the receiver's public key; later the receiver may send a token, which is generated based on the receiver's private key, to the server so that the latter can search over the tags attached to each encrypted message. Non-interactive PEKS is also referred to as searchable encryption in [6]. As a motivation example, Boneh *et al.* [6] show that non-interactive PEKS can be used for routing emails in an email system (more details are in Section 2.2). Among other applications, Waters *et al.* [30] show that non-interactive PEKS can be used to build an encrypted and searchable audit log.

Related Work. Abdalla *et al.* [1] study the issue of consistency for non-interactive PEKS, provide a transform of an anonymous IBE scheme to a secure non-interactive PEKS scheme, and propose three extensions, namely anonymous HIBE, public-key encryption with temporary keyword search, and identity-based encryption with keyword search. Di Crescenzo and Saraswat [10] propose a non-interactive PEKS construction based on Jacobi symbols. Khader [16] shows how to construct non-interactive PEKS based on K-Resilient IBE. Baek, Safavi-Naini, and Susilo [3] discuss the issues about refreshing keywords, avoiding the secure channel for protecting trapdoors, and searching on multiple keywords.

There are a number of extentions to the concept of non-interactive PEKS. In the original definition, the search is only done by comparing the equality of the keywords contained in the token and tags. Boneh and Waters [7] extend non-interactive PEKS to support conjunctive, subset, and range comparisons over the keywords. Hwang and Lee [15] investigate non-interactive PEKS in multiuser setting, where there are m receivers in the scheme and the m public keys are used to encrypt the keywords. Implicitly assumed in [6], the public-key encryption scheme and the non-interactive PEKS scheme share the same key pair but none of the security definitions for these primitives has taken this into account. The authors in [2, 20] investigate hybrid models for combining public-key encryption and non-interactive PEKS, where both primitives share the same public/private key pair.

Besides these follow-ups, non-interactive PEKS is also related to the information retrieval problem in the private database setting and public database setting. In the private database setting, a user wishes to upload its private data to a remote database and wishes to keep the data private from the database. Later, the user must be able to retrieve from the remote database all records that contain a particular keyword. Goldreich and Ostrovsky [14] first propose solutions to this problem, and many follow-ups exist (e.g. [4, 8, 11, 13, 19]). In this setting, the user stores the encrypted data at the server, hence, it is different from the case of non-interactive PEKS. In the public setting, a user wants to retrieve data from a database, which stores data in plaintext, and keep the index of the retrieved data private from the database. Public Information Retrieval (PIR) protocols, proposed by Chor *et al.* [9], are solutions to this problem. Gasarch [12] provides a very detailed summary of PIR protocols and lower/upper bounds on communication complexity, and Ostrovsky and Skeith III [17] also provides a summary. In this setting, the database stores data in plaintext, hence, it is also different from the case of non-interactive PEKS.

Our Contribution. We investigate a non-interactive PEKS application proposed by Boneh *et al.* [6] and show a number of observations. We show that the security definitions of non-interactive PEKS given in [6] do not match the adversaries in practice, namely a curious server and a curious message sender. As a result, the proposed PEKS application cannot provide enough privacy protection for the

receiver in practice.

As a solution, we propose the notion of interactive PEKS, which, in contrast to non-interactive PEKS, requires the tag to be generated interactively by the message sender and the receiver. For this new primitive, we identify two types of adversaries, namely a curious server and a curious message sender. We provide security formulations for the desirable properties, namely soundness, consistency, and semantic securities against both types of adversaries. We propose a construction for interactive PEKS which is derived from the pairing-based non-interactive PEKS scheme given in [6]. The proposed scheme achieves semantic security against the a curious server in the random model, and achieves semantic security against the a curious message sender based on the BDH assumption in the random model.

Organization. The rest of the paper is organized as follows. In Section 2 we review the concept of non-interactive PEKS and show that a non-interactive PEKS scheme cannot provide necessary privacy protection for the receiver. In Section 3 we introduce the concept of interactive PEKS and provide the security definitions. In Section 4 we propose a construction for interactive PEKS and prove its security. We also provide some further remarks on the definition and construction of interactive PEKS. In Section 5 we conclude the paper.

2 Review the Concept of PEKS

In this section we review the concept of non-interactive PEKS and present our observations on the related security concerns through a case study.

2.1 Current Security Model for Non-interactive PEKS

A non-interactive PEKS scheme mainly involves the following entities: message senders, a receiver, and a server. Formally, a non-interactive PEKS scheme consists of the following polynomial time randomized algorithms:

- $\mathsf{KeyGen}(k)$: Run by the receiver, this algorithm takes a security parameter k as input and generates a public/private key pair (A_{pub}, A_{priv}).
- $\mathsf{PEKS}(A_{pub}, W)$: Run by a message sender, this algorithm takes A_{pub} and a keyword W as input and outputs a tag S for W.
- $\mathsf{Trapdoor}(A_{priv}, W)$: Run by the receiver, this algorithm takes A_{priv} and a keyword W as input and outputs a trapdoor T_W.
- $\mathsf{Test}(A_{pub}, S, T_{W'})$: Run by the server, this algorithm takes A_{pub}, S, and $T_{W'}$ as input, where

$$S = \mathsf{PEKS}(A_{pub}, W),\ T_{W'} = \mathsf{Trapdoor}(A_{priv}, W'),$$

and outputs 1 if $W = W'$ and 0 otherwise.

With a non-interactive PEKS scheme, the workflow of the underlying application consists of two types of operations. More details can be found in Section 2.2.

1. The first operation is that, a message sender encrypts his message, runs PEKS to generate some tags (encrypted keywords) for the message, and reposits the ciphertext and the tags at the server. It is assumed that a tag is generated for each keyword.

2. The second operation is that, the receiver runs $\mathsf{Trapdoor}$ to generate a trapdoor for a certain keyword, sends the trapdoor to the server which will run Test to search over the tags attached to each encrypted message.

Analogous to the case of public key encryption [5], for non-interactive PEKS, the semantic security against an adaptive adversary is evaluated by the following game (as depicted in Figure 1) between a challenger and an adversary $\mathcal{A}$.

1. The challenger runs the $\mathsf{KeyGen}(k)$ to generate A_{pub} and A_{priv}. It gives A_{pub} to the adversary.

2. The adversary can adaptively ask the challenger for the trapdoor T_W for any keyword $W \in \{0,1\}^*$ of his choice.

3. At some point, the adversary $\mathcal{A}$ sends the challenger two words W_0, W_1 on which it wishes to be challenged. The only restriction is that the adversary did not previously ask for the trapdoors T_{W_0} and T_{W_1}. The challenger picks a random bit $b \in \{0,1\}$ and gives the adversary the challenge $S_b = \mathsf{PEKS}(A_{pub}, W_b)$.

4. The adversary can continue to ask for the trapdoor T_W for any keyword W of his choice as long as it is not W_0 and W_1.

5. Eventually, the adversary $\mathcal{A}$ outputs b'.

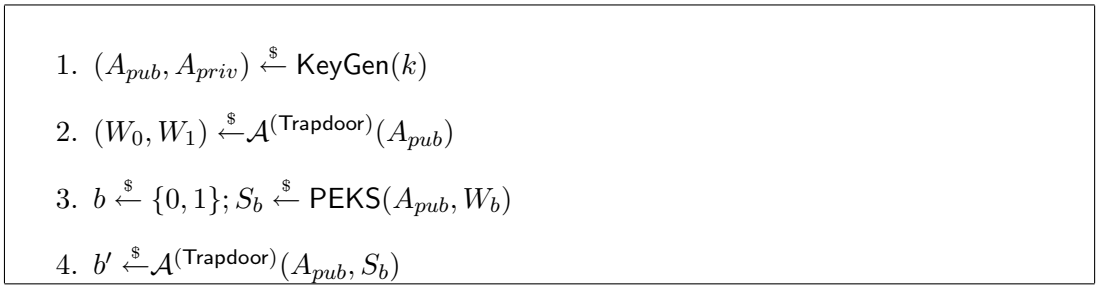

Figure 1: Semantic Security of Non-interactive PEKS

Definition 1. *A non-interactive PEKS scheme is semantically secure if any polynomial time adversary has only a negligible advantage in the above game, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

If a non-interactive PEKS scheme achieves semantic security, it means that the adversary cannot determine whether or not two tags contain the same keyword without knowing the related trapdoors. Boneh *et al.* [6] show that a semantically secure non-interactive PEKS scheme implies a chosen ciphertext secure identity-based public key encryption scheme. However, we note that the adversary (as defined in the above game) does not correspond to any type of "real adversary" against a non-interactive PEKS scheme in practice. We demonstrate this in the following subsection and formally address this issue in Section 3.

Besides the semantic security, soundness and consistency are also important concerns. The soundness attribute says that the value of $\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W), T_{W'})$ should should be 1 if $W = W'$, and the consistency attribute says that the value of $\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W), T_{W'})$ should be 0 if $W \neq W'$. The consistency attribute of PEKS has been extensively studied by Abdalla *et al.* [1]. We skip the details here and present the definitions in Section 3.2.

2.2 Observations on Non-interactive PEKS

We first restate the email routing example, which has been used as the motivation for non-interactive PEKS by Boneh *et al.* [6], and then describe our observations.

> *Suppose that, in an email system, for a user Alice, her emails are created by various people and encrypted using her public key. Suppose also that some keywords are encrypted and attached to Alice's emails. The main goal is to enable Alice to give the email server the ability to test whether a certain keyword has been attached to the email, but the server should learn nothing else about the email.*

With a PEKS scheme, Boneh *et al.* propose the following solution. Alice generates a pair (A_{pub}, A_{priv}). If Bob wants to send a message M with keywords $W_1, W_2, \cdots, W_n$ to Alice, he sends

$$\mathsf{Encrypt}(A_{pub}, M), \mathsf{PEKS}(A_{pub}, W_1), \mathsf{PEKS}(A_{pub}, W_2), \cdots, \mathsf{PEKS}(A_{pub}, W_n).$$

If Alice wants to retrieve all the emails containing the keyword W, she gives the email server a trapdoor T_W. According to the definition of PEKS, given $\mathsf{PEKS}(A_{pub}, W_i)$ and T_W the server can test whether $W_i = W$. Hence, the email server can identify all the emails which contain the keyword W and route them later.

With respect to this solution, Alice needs to publish all her keywords. Note that, for an email system, we assume Alice does not know Bob in advance. If Alice does not publish her keywords, then Bob cannot attach a useful tag that can be used by

the server to perform the test. Furthermore, it is reasonable to generally assume that the server publishes a polynomial number of keywords in non-interactive PEKS shcemes.

Based on the above observation, we have the following security concerns.

1. In the above solution, it is unclear whether or not the privacy of message M is achieved even if the public-key encryption scheme is secure. The reason is that both primitives, namely the public-key encryption scheme and the PEKS scheme, share the same key pair but none of the security definitions for these primitives has taken this into account. In fact, this problem has motivated the formulation of hybrid models in [2, 20].

2. Given a trapdoor, the server can recover the corresponding keyword based on the assumption that the keyword set is public and polynomial size in the security parameter. To do this, the server can generate a tag for each keyword and test it with the trapdoor at hand. This means there is no privacy over the keywords contained in her emails against a curious server. We can conclude that the above solution can only provide very limited privacy guarantee for the receiver, especially against a curious server.

3. There might be an inference attack in this solution. Suppose a message sender Eve has reposited $\mathsf{Encrypt}(A_{pub}, M), \mathsf{PEKS}(A_{pub}, W)$ at the server. If he notices the receiver has retrieved his message and another message from Bob, then he can determine that Bob has sent a message containing the same keyword W. As a result, we should assume there is a secure link between the server and the message receiver for the retrieval process (the transmission of both the trapdoors and retrieved messages). Note that the protection of trapdoor is not enough here, which means that the proposal of Baek, Safavi-Naini, and Susilo [3] might also suffer from the inference attack.

In this context of email routing, the security definitions of non-interactive PEKS given in Section 2.1 do not match the possible adversaries, namely a curious server and a curious message sender. As a result, the proposed solution cannot provide enough privacy protection for the receiver in practice.

3 The Concept of Interactive PEKS

In this section we first introduce the concept of interactive PEKS and then present the formal security definitions.

3.1 Introduction to the Concept

Just the same as in the case of non-interactive PEKS, an interactive PEKS scheme also involves the following entities: message senders, a receiver, and a server. The

observations in Section 2.2 show that, given a trapdoor, the server can run PEKS to generate a tag for each keyword and then run Test to find out the keyword related to the trapdoor. In order to protect the privacy for the receiver against a curious server, the server should not be able to generate the trapdoor by itself. One possible way to achieve this goal is making PEKS an interactive algorithm which is run between a message sender and the receiver, such that a message sender cannot compute a meaningful tag by himself. Hence, we propose a primitive called interactive PEKS, which is different from the non-interactive PEKS in that the algorithm PEKS is an interactive one.

Formally, an interactive PEKS scheme consists of the following polynomial time randomized algorithms:

- $\mathsf{KeyGen}(k)$: Run by the receiver, this algorithm takes a security parameter k as input and generates a public/private key pair (A_{pub}, A_{priv}). We assume that the information about the keyword set is included in A_{pub}.
- $\mathsf{PEKS}(A_{pub}, W; A_{priv})$: Run between a message sender and the receiver, this algorithm takes A_{pub} and a keyword W from the message sender and A_{priv} from the receiver as input, and outputs a tag S for W.
- $\mathsf{Trapdoor}(A_{priv}, W)$: Run by the receiver, this algorithm takes A_{priv} and a keyword W as input and outputs a PEKS trapdoor T_W.
- $\mathsf{Test}(A_{pub}, S, T_{W'})$: Run by the server, this algorithm takes A_{pub}, S, and $T_{W'}$ as input, where

  $$S = \mathsf{PEKS}(A_{pub}, W),\ T_{W'} = \mathsf{Trapdoor}(A_{priv}, W'),$$

  and outputs 1 if $W = W'$ and 0 otherwise.

The algorithm PEKS can be considered as a secure two-party function between a message sender and the receiver. Compared with the original definition, the only difference here is that PEKS is an interactive algorithm between a message sender and the receiver.

Before describing the security definitions for interactive PEKS, we first describe our assumptions.

1. Due to the observation in Section 2.2, we assume the key pair (A_{pub}, A_{priv}) is only used for the interactive PEKS but not for general public-key encryption services. In other words, the receiver should should employ another key pair to achieve the general encryption service. Compared with the hybrid models in [2, 20], a separate discussion for these primitives has many advantages. For example, we can study both primitives separately and design solutions with them using a modular approach.

2. The public keyword set is $\mathcal{W}$ of cardinality N, where $N \geq 2$ is an integer and every keyword is a binary string. Without loss of generality, we can further assume N is a polynomial in the security parameter k.

3. The communication link between the receiver and the server is secure in both confidentiality and integrity. This assumption is essential to prevent inference attacks against curious message senders. In fact, this kind of assumption is widely adopted in other types of protocols. For example, in the multi-database PIR protocols [9], it is implicitly assumed that a curious database cannot observe the communication between the user and any other database.

4. The server cannot be a message sender and there is no collusion between message senders and the server. The server should delete the tokens immediately after the Test operation, but it does not need to keep the tags (and the encrypted messages) private.

3.2 Security Definitions

Soundness and consistency. The soundness attribute says that, for any keywords W and W', the value of $\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W), T_{W'})$ should be 1 if $W = W'$.

Definition 2. *An interactive PEKS scheme is sound if the probability $\mathcal{P}$ is negligible, where*

$$\mathcal{P} = \max_{W} \Pr[\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W; A_{priv}), \mathsf{Trapdoor}(A_{priv}, W)) = 0]$$

The consistency attribute says that, for any keywords W and W', the value of $\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W), T_{W'})$ should be 0 if $W \neq W'$.

Definition 3. *An interactive PEKS scheme is computationally consistent if the probability $\mathcal{P}$ is negligible, where*

$$\mathcal{P} = \max_{W \neq W'} \Pr[\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W; A_{priv}), \mathsf{Trapdoor}(A_{priv}, W')) = 1]$$

The main functionality of an interactive PEKS scheme is to enable message senders to reposit encrypted messages and related tags at the server, and the receiver can retrieve a certain encrypted message by asking the server to search on the tags attached to each encrypted message. We identify two types of adversaries, namely a curious server and a curious message sender.

Semantic security against a curious server. A curious server has access to the encrypted messages and the related tags from message senders, as well as the trapdoors from the receiver. The main aim of this type of adversary is to learn some information about the keywords contained in the tags and tokens attached to some encrypted message. Since the adversary can run Test to match the keywords contained in a tag and a token, it is sufficient to model the security by evaluating the adversary's capability to distinguish the encrypted keywords in tags only.

Note that in practice, for the server, the distribution of keywords in the received tags might not be uniform at random and so is for the distribution of keywords in the received tokens. However, for the simplicity of discussion, we assume that the keyword distribution is uniform in the tag generation and the token generation processes.

The attack game for semantic security is depicted in Figure 2. For the PEKS oracle, the adversary does not need to provide any input, the challenger samples a keyword uniformly at random and returns $\mathsf{PEKS}(A_{pub}, W; A_{priv})$. For the $\mathsf{Trapdoor}$ oracle, the adversary does not need to provide any input, the challenger samples a keyword uniformly at random and returns the trapdoor $\mathsf{Trapdoor}(A_{priv}, W)$. In contrast to the definition for non-interactive PEKS, as shown in Figure 1, W_0 and W_1 are allowed to be issued to $\mathsf{Trapdoor}$ in this definition.

1. $(A_{pub}, A_{priv}) \xleftarrow{\$} \mathsf{KeyGen}(k)$
2. $(W_0, W_1) \xleftarrow{\$} \mathcal{A}^{(\mathsf{PEKS},\mathsf{Trapdoor})}(A_{pub})$
3. $b \xleftarrow{\$} \{0,1\}; S_b \xleftarrow{\$} \mathsf{PEKS}(A_{pub}, W_b; A_{priv})$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{PEKS},\mathsf{Trapdoor})}(A_{pub}, S_b)$

Figure 2: Semantic security against a curious server

Definition 4. *An interactive PEKS scheme achieves semantic security against a curious server if any polynomial time adversary has only a negligible advantage in the above game, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

Semantic security against a curious message sender. We consider a general environment where the communication link between a message sender and the server may not be secured and the server may keep all encrypted messages and the related tags in public. Under this assumption, a curious message sender has access to the encrypted messages and the related tags from all message senders, as well as the information gained from the tag generation process, i.e. the executions of PEKS. The main aim of this type of adversary is also to learn some information about the

keywords contained in the tags attached to some encrypted message. Compared with the a curious server, here the adversary does not have access to any token.

The attack game is depicted in Figure 3. For the PEKS oracle, the adversary chooses a keyword at its will and runs PEKS interactively with the challenger to compute $\mathsf{PEKS}(A_{pub}, W; A_{priv})$. However, in generating the challenge S_b, the challenger chooses W_b and does the computation alone. In contrast to the definition for non-interactive PEKS, as shown in Figure 1, the adversary has no access to the $\mathsf{Trapdoor}$ oracle in this definition.

1. $(A_{pub}, A_{priv}) \xleftarrow{\$} \mathsf{KeyGen}(k)$
2. $(W_0, W_1) \xleftarrow{\$} \mathcal{A}^{(\mathsf{PEKS})}(A_{pub})$
3. $b \xleftarrow{\$} \{0,1\}; S_b \xleftarrow{\$} \mathsf{PEKS}(A_{pub}, W_b; A_{priv})$
4. $b' \xleftarrow{\$} \mathcal{A}^{(\mathsf{PEKS})}(A_{pub}, S_b)$

Figure 3: Semantic security against a curious message sender

Definition 5. *An interactive PEKS scheme achieves semantic security against a curious message sender if any polynomial time adversary has only a negligible advantage in the above game, where the advantage is defined to be* $|\Pr[b' = b] - \frac{1}{2}|$.

4 A Construction of Interactive PEKS

In this section we propose a pairing-based interactive PEKS scheme and prove its security.

4.1 A Preliminary of Pairing

We review the necessary knowledge about pairing and the related assumptions. More detailed information can be found in the seminal paper [6]. A pairing (or, bilinear map) satisfies the following properties:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative groups of prime order p;
2. g is a generator of $\mathbb{G}$;
3. $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an efficiently-computable bilinear map with the following properties:
   - Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.

- Non-degenerate: $\hat{e}(g, g) \neq 1$.

As defined in [6], $\mathbb{G}$ is said to be a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_1$ and an efficiently-computable bilinear map $\hat{e}$ as above.

The Bilinear Diffie-Hellman (BDH) problem in $\mathbb{G}$ is as follows: given a tuple $g, g^a, g^b, g^c \in \mathbb{G}$ as input, output $\hat{e}(g, g)^{abc} \in \mathbb{G}_1$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(g, g^a, g^b, g^c) = \hat{e}(g, g)^{abc}] \geq \epsilon.$$

Similarly, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the decision BDH problem in $\mathbb{G}$ if

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^b, g^c, T) = 0]| \geq \epsilon.$$

where the probability is over the random choice of $a, b, c \in \mathbb{Z}_p^*$, the random choice of $T \in \mathbb{G}_1$, and the random bits of $\mathcal{A}$.

Definition 6. *We say that the (decision) (t, ϵ)-BDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (decision) BDH problem in $\mathbb{G}$.*

Given a security parameter k, a problem (say, BDH) is said to be intractable if any adversary has only negligible advantage in reasonable time. We usually define a scheme to be secure if any adversary has only a negligible advantage in the underlying security model. The time parameter is usually ignored.

Definition 7. *The function $P(k) : \mathbb{Z} \to \mathbb{R}$ is said to be negligible if, for every polynomial $f(k)$, there exists an integer N_f such that $P(k) \leq \frac{1}{f(k)}$ for all $k \geq N_f$.*

4.2 Proposed Construction of Interactive PEKS

We construct an interactive PEKS scheme based on pairing. Our construction is derived from the non-interactive PEKS scheme by Boneh *et al.* [6]. The algorithms are defined as follows.

- $\mathsf{KeyGen}(k)$: This algorithm generates two cyclic groups $\mathbb{G}$ and $\mathbb{G}_1$ of prime order p, a generator g of $\mathbb{G}$, a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, $\alpha, \beta \in_R \mathbb{Z}_p^*$, and two hash functions $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathsf{H}_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. The public key is $A_{pub} = (\mathbb{G}, \mathbb{G}_1, p, g, \mathsf{H}_1, \mathsf{H}_2, \hat{e}, g^\alpha)$ and the private key is $A_{priv} = (\alpha, \beta)$.
- $\mathsf{PEKS}(A_{pub}, W; A_{priv})$: This interactive algorithm works as follows:
  1. The message sender selects $r \in_R \mathbb{Z}_p^*$ and sends (g^r, W) to the receiver as a request.

2. After receiving (g^r, W), the receiver sends S_2' to the message sender, where

$$S_2' = \hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)}).$$

3. After receiving S_2', the message sender outputs a tag S for W, where $S_1 = g^r,\ S_2'' = \hat{e}(\mathsf{H}_1(W), g^\alpha)^r,\ S = (S_1, S_2)$.

$$\begin{aligned} S_2 &= S_2' \cdot S_2'' \\ &= \hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)}) \cdot \hat{e}(\mathsf{H}_1(W), g^\alpha)^r \\ &= \hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)} \cdot \mathsf{H}_1(W)^\alpha) \end{aligned}$$

- $\mathsf{Trapdoor}(A_{priv}, W)$: This algorithm outputs a trapdoor T_W for W, where $T_W = \mathsf{H}_1(W)^\alpha \cdot g^{\mathsf{H}_2(W||\beta)}$.
- $\mathsf{Test}(A_{pub}, S, T_{W'})$: With the input $A_{pub}, S, T_{W'}$, where $S_1 = g^r$,

$$S_2 = \hat{e}(\mathsf{H}_1(W), g^\alpha)^r \cdot \hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)}),\ T_{W'} = \mathsf{H}_1(W')^\alpha \cdot g^{\mathsf{H}_2(W'||\beta)},$$

this algorithm outputs 1 if $S_2 = \hat{e}(S_1, T_{W'})$ and 0 otherwise.

From the above description, the following equation holds for any W.

$$\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W; A_{priv}), \mathsf{Trapdoor}(A_{priv}, W)) = 1$$

Therefore, the scheme is sound unconditionally.

Lemma 1. *The proposed scheme is consistent in the random oracle model.*

Proof sketch. For any $W \neq W'$, the following equation

$$\mathsf{Test}(A_{pub}, \mathsf{PEKS}(A_{pub}, W; A_{priv}), \mathsf{Trapdoor}(A_{priv}, W')) = 1$$

means

$$\hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)} \cdot \mathsf{H}_1(W)^\alpha) = \hat{e}(g^r, \mathsf{H}_1(W')^\alpha \cdot g^{\mathsf{H}_2(W'||\beta)}) \text{ for } r \in_R \mathbb{Z}_q^*.$$

Since H_1 and H_2 are modeled as random oracles, the probability that the above equation holds is $\frac{1}{q-1}$. Since $\frac{1}{q-1}$ is negligible, the lemma holds. □

Lemma 2. *The proposed scheme achieves semantic security against a curious server in the random oracle model.*

Proof sketch. Suppose an adversary $\mathcal{A}$ has the non-negligible advantage ϵ in the attack game depicted in Figure 2. The security proof is done through a sequence of games [18].

Game_0: In this game, the challenger faithfully simulates the protocol execution and answers the oracle queries from $\mathcal{A}$. We assume the challenger simulates the random oracles as follows. For H_1, the challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}$ (the hash-code for this message), and an element of $\mathbb{Z}_p^*$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}$; otherwise, the challenger returns g^y, where y a randomly chosen element of $\mathbb{Z}_p^*$, and stores the new vector in the list. For H_2, the challenger maintains a list of vectors, each of them containing a request message and an element of $\mathbb{Z}_p^*$ (the hash-code for this message). After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{Z}_p^*$; otherwise, the challenger returns u which is a randomly chosen element of $\mathbb{Z}_p^*$, and stores the new vector in the list. Let $\delta_0 = \Pr[b' = b]$, as we assumed at the beginning, $|\delta_0 - \frac{1}{2}| = \epsilon$.

Game_1: In this game, the challenger performs in the same way as in Game_0 except for the following: the challenger selects W_0', W_1' before starting the game and aborts if $W_0' \neq W_0$ or $W_1' \neq W_1$ in the game. Since the keyword set has the cardinality N, the probability that the challenger does not abort is $\frac{1}{N(N-1)}$. Let $\delta_1 = \Pr[b' = b]$ given that the challenger successfully ends, in which case $\delta_1 = \delta_0$. Let θ_1 be the probability that the challenger successfully ends and $b' = b$ in Game_1, then we have $\theta_1 = \frac{\delta_1}{N(N-1)}$.

Game_2: In this game, the challenger performs in the same way as in Game_1 except for answering the following two types of oracles in Step 2 and 4 of the game (depicted in Figure 2).

1. PEKS: The challenger first samples W. If $W = W_0'$, the challenger returns $\mathsf{PEKS}(A_{pub}, W_1'; A_{priv})$; if $W = W_1'$, the challenger returns $\mathsf{PEKS}(A_{pub}, W_0'; A_{priv})$.

2. $\mathsf{Trapdoor}$: The challenger first samples W. If $W = W_0'$, the challenger returns $\mathsf{Trapdoor}(A_{priv}, W_1')$; if $W = W_1'$, the challenger returns $\mathsf{Trapdoor}(A_{priv}, W_0')$.

The game Game_2 is identical to Game_1 except that the event E_1 occurs: the adversary queries H_2 with the input $*||\beta$ or $*||\beta$, where $*$ is any string. Since H_1 is modeled as a random oracle, the probability $\Pr[E_1]$ is negligible because β is a secret. When E_1 does not occur, let $\delta_2 = \Pr[b' = b]$ given that the challenger successfully ends and θ_2 be the probability that the challenger successfully ends and $b' = b$ in Game_2. We have $\theta_2 = \frac{\delta_2}{N(N-1)}$. Since Game_2 only differs from Game_1 when E_1 occurs, then we have $|\theta_2 - \theta_1| \leq \Pr[E_1]$, i.e. $|\frac{\delta_2}{N(N-1)} - \frac{\delta_1}{N(N-1)}| \leq \Pr[E_1]$. On the other hand, if E_1 does not occur, we have $\delta_2 = 1 - \delta_1$. As a result, we have $|\frac{1-\delta_1}{N(N-1)} - \frac{\delta_1}{N(N-1)}| \leq \Pr[E_1]$, so that $|\delta_0 - \frac{1}{2}| \leq \frac{1}{2}N(N-1)\Pr[E_1]$, i.e. $\epsilon \leq \frac{1}{2}N(N-1)\Pr[E_1]$. Since N is a polynomial in the security parameter k and $\Pr[E_1]$ is negligible, the lemma now follows. $\square$

Compared with the original scheme in [6], the receiver has an additional secret β in the proposed scheme. The computations involved with this secret make the secheme secure against a curious server. By removing these computations, we will obtain the original scheme in [6]. On the other hand, the adversary has less privilege in the semantic security definition against a curious message sender than in the original definition, without considering β. From the security proof by Boneh *et al.* (i.e. Theorem 3.1 in [6]) we immediately get the following lemma.

Lemma 3. *The proposed scheme achieves semantic security against a curious message sender based on the BDH assumption assumption in the random oracle model.*

4.3 Some Remarks on Interactive PEKS

According to our definition for the algorithm PEKS in Section 3.1, the receiver is required to be online to for each execution. This is certainly a burden for the receiver in practice. In practice, this requirement can be relaxed in many ways. For example,

1. One way is that the receiver can ask every message sender to keep a copy of β. With this method, if a message sender is compromised then the security is compromised.

2. Another way is that the receiver can find a trusted third party and stores β and the message sender interactively executes the algorithm with it.

In addition, for every keyword, it may be possible that the message sender to run PEKS only once with the receiver and regenerate a new tag later by itself. Consider the proposed scheme in Section 4.2. For any keyword W, the tag is $S = (S_1, S_2)$, where $S_1 = g^r$, $S_2 = \hat{e}(g^r, g^{\mathsf{H}_2(W||\beta)} \cdot \mathsf{H}_1(W)^\alpha)$ for some $r \in \mathbb{Z}_p^*$. With S, the message sender can generate a new tag for W as $S' = S^{r'} = (S_1^{r'}, S_2^{r'})$ where $r \in_R \mathbb{Z}_p^*$. The outcome is exactly the same as in the case where the message sender runs PEKS with the receiver.

When introducing the concept of interactive PEKS in Section 3.1, we explicitly make the following assumption: *The server cannot be a message sender and there is no collusion between message senders and the server.* In practice, this assumption may be unrealistic in some applications, in which case we can generally assume that the server colludes with message senders U_i $(1 \leq i \leq n)$. Informally, the semantic security definitions may be extended in the following way to cope with the situation.

1. Suppose the colluded message senders U_i $(1 \leq i \leq n)$ run PEKS with the receiver for keyword set $\mathcal{W}'$. The semantic security against the a curious server is identical to that defined in Figure 2, except for an additional restriction that $W_0, W_1 \notin \mathcal{W}'$.

2. For the semantic security against colluded message senders U_i $(1 \leq i \leq n)$, the semantic security definition is the same as in the above case for the a curious server (as they are colluded). For a curious message sender apart from the colluded message senders U_i $(1 \leq i \leq n)$, the semantic security definition is identical to that described in Figure 3.

5 Conclusion

In this paper we have shown that the security definitions for non-interactive PEKS do not match the adversaries in the practical applications. We have proposed the concept of interactive PEKS as an alternative, categorized the possible adversaries (namely a curious user and a curious server), and provided the necessary security definitions. We have also proposed a construction for interactive PEKS and proven its security in the proposed security model. In practice, the online tag generation may be a burden for the underlying application, nonetheless, we have pointed out that a number of methods can be employed to overcome this problem. Note that we have only defined the semantic securities against a chosen plaintext attack, it is an interesting future work to define the semantic securities against a chosen ciphertext attack and provide corresponding protocol constructions.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In V. Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] J. Baek, R. Safavi-Naini, and W. Susilo. On the integration of public key data encryption and public key encryption with keyword search. In S. K. Katsikas, J. Lopez, M. Backes, S. Gritzalis, and B. Preneel, editors, *Information Security, 9th International Conference, ISC 2006*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2006.

[3] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In O. Gervasi, B. Murgante, A. Laganà, D. Taniar, Y. Mun, and M. L. Gavrilova, editors, *Computational Science and Its Applications - ICCSA 2008, International Conference*, volume 5072 of *Lecture Notes in Computer Science*, pages 1249–1259. Springer, 2008.

[4] F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In L. Chen, Y. Mu, and W. Susilo, editors, *Information*

Security Practice and Experience, 4th International Conference, ISPEC 2008, volume 4991 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 2008.

[5] 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 *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[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 *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[7] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[8] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455, 2005.

[9] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[10] G. Di Crescenzo and V. Saraswat. Public key encryption with searchable keywords based on jacobi symbols. In K. Srinathan, C. P. Rangan, and M. Yung, editors, *Progress in Cryptology - INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2007.

[11] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 79–88. ACM, 2006.

[12] W. Gasarch. A survey on private information retrieval. http://www.cs.umd.edu/ gasarch/pir/pir.html.

[13] E. Goh. Secure Indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[14] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[15] Y. H. Hwang and P. J. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007, First International Conference*, volume 4575 of *Lecture Notes in Computer Science*, pages 2–22. Springer, 2007.

[16] D. Khader. Public key encryption with keyword search based on k-resilient ibe. In O. Gervasi and M. L. Gavrilova, editors, *Computational Science and Its Applications - ICCSA 2007, International Conference*, volume 4707 of *Lecture Notes in Computer Science*, pages 1086–1095. Springer, 2007.

[17] R. Ostrovsky and W. E. Skeith III. A survey of single database PIR: Techniques and applications. Cryptology ePrint Archive: Report 2007/059, 2007.

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

[19] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[20] 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 *Lecture Notes in Computer Science*, pages 159–174. Springer, 2007.