J Med Syst (2015) 39: 11
DOI 10.1007/s10916-014-0178-y

Efficient Secure-Channel Free Public Key Encryption with Keyword Search for EMRs in Cloud Storage

Lifeng Guo · Wei-Chuen Yau

Received: 9 August 2014 / Accepted: 27 November 2014 / Published online: 30 January 2015

Abstract Searchable encryption is an important cryptographic primitive that enables privacy-preserving keyword search on encrypted electronic medical records (EMRs) in cloud storage. Efficiency of such searchable encryption in a medical cloud storage system is very crucial as it involves client platforms such as smartphones or tablets that only have constrained computing power and resources. In this paper, we propose an efficient secure-channel free public key encryption with keyword search (SCF-PEKS) scheme that is proven secure in the standard model. We show that our SCF-PEKS scheme is not only secure against chosen keyword and ciphertext attacks (IND-SCF-CKCA), but also secure against keyword guessing attacks (IND-KGA). Furthermore, our proposed scheme is more efficient than other recent SCF-PEKS schemes in the literature.

Keywords Electronic medical record · Cloud storage · Public key encryption · Keyword search · Keyword guessing attack · Standard model

Introduction

With the increasing popularity of adopting cloud technologies, many health care providers tend to store electronic medical records (EMRs) in cloud storage [10–13, 18, 24]. Health care practitioners can enjoy the benefit of accessing medical records from anywhere with internet connection. To protect privacy of the data, health care practitioners may need to encrypt their data before storing in cloud storage. There should be a mechanism for them to search on the encrypted data without compromising the privacy of patients [9]. In 2004, Boneh et al. [4] introduced the notion of public key encryption with keyword search (PEKS). A PEKS allows one to perform encrypted keyword search on ciphertexts without revealing the original message. This notion has many useful applications, for example, email routing [3, 4], cloud storage [22], electronic health record systems [31, 32], etc.

A PEKS scheme requires a secure channel to transmit a trapdoor from a receiver to a server. Otherwise, an attacker may easily identify which encrypted messages are related to the given trapdoor. Baek et al. [5] solved the secure channel problem by proposing a secure-channel free PEKS (SCF-PEKS) which encrypts keyword with both server's and receiver's public keys. This ensures that only a designated server can perform the search and prevents other party that is without server's private key from determining the relation between keyword ciphertexts and trapdoors. An SCF-PEKS scheme is also known as a PEKS scheme with a designated tester (dPEKS) [27–30]. Baek et al.'s SCF-PEKS scheme [5] is proven secure in the random oracle model. Fang et al. [19] later proposed an SCF-PEKS scheme which is secure in the standard model.

This article is part of the Topical Collection on *Systems-Level Quality Improvement*.

L. Guo
School of Computer and Information Technology, Shanxi University, Taiyuan, 030006, Shanxi, People's Republic of China
e-mail: lfguo@sxu.edu.cn

W.-C. Yau (✉)
Faculty of Engineering, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor Darul Ehsan, Malaysia
e-mail: wcyau@mmu.edu.my

The problem of off-line keyword guessing attacks (KGA) was first addressed by Byun et al. [7]. Subsequently, Yau et al. [35] also showed KGA on Baek et al.'s SCF-PEKS scheme [5]. This attack works as the keyword space of keywords used in PEKS and SCF-PEKS schemes are small. Attackers exploit this weakness by exhaustively guessing some candidate keywords and check whether their guesses are correct or not. Rhee et al. [28] enhanced the security model of [5] as well as proposed the concept of trapdoor indistinguishability. They showed that trapdoor indistinguishability is a sufficient condition for ensuring the security against off-line keyword guessing attacks.

Emura et al. [15, 16] showed a generic construction of adaptive SCF-PEKS from anonymous identity-based encryption (IBE) . Subsequently, Emura and Rahman extended the work in [15] and constructed a more efficient adaptive SCF-PEKS using IBE with partitioned ciphertext structure [16, 17]. However, the schemes proposed in [15–17] are not secure against keyword guessing attacks. Rhee et al. [29] also proposed two generic transformations to construct an SCF-PEKS scheme using two IBE schemes that are either combined in parallel or in sequence. Very recently, Fang et al. [20] proposed the strongest model in SCF-PEKS that is secure against chosen keyword and ciphertext attack (IND-SCF-CKCA) and against keyword guessing attack (IND-KGA). At the same time, they constructed an SCF-PEKS scheme which is proven secure in the standard model. The construction of Fang et al.'s SCF-PEKS scheme [20] is based on Gentry's IBE scheme [21]. They used the method of identity's construction in Waters's IBE scheme [34] in order to achieve KGA resiliency. They also include the test query in their security model which makes it stronger than other SCF-PEKS's security model in the literature. To secure against this security model, they applied strongly unforgeable one-time signatures on the ciphertext elements of their SCF-PEKS scheme.

Our contributions In this paper, we propose a very efficient SCF-PEKS scheme that is secure against chosen keyword and ciphertext attack (IND-SCF-CKCA) and against off-line keyword guessing attacks (IND-KGA) without random oracles. Fang et al.'s SCF-PEKS scheme [20] applied strongly unforgeable one-time signature to secure against IND-SCF-CKCA. However, our proposed SCF-PEKS scheme uses the technique of [23] to resist the same attack but requires less computation overhead as well as shorter ciphertext length as compared to the SCF-PEKS scheme of [20]. The main technique used by Fang et al. [20] to resist off-line keyword guessing attacks (IND-KGA) is to use Waters's hash function [34] to construct user's key that protects the anonymity of keyword in the trapdoor. This, however, results large key size for user's public and private keys. In contrast, we adopt the method of generating randomness for trapdoor to achieve IND-KGA secure as well as greatly reduce the key size of the proposed SCF-PEKS scheme. In terms of security proof, Fang et al.'s security proof [20] depends on a stronger q-ABDHE assumption which is related to the number of private key generation queries made by the adversary. While, the security proof of our proposed SCF-PEKS scheme is based on a weaker QDBDH assumption which does not has this constraint. This paper answers in the affirmative the question posed by Fang et al. [20] on how to construct a more efficient SCF-PEKS scheme without random oracles. Our proposed efficient and secure SCF-PEKS scheme is suitable to be used in medical cloud storage to protect the privacy of patients' medical records.

Paper organization The rest of this paper is organized as follows: Section "Preliminaries" reviews the definitions related to our proposed SCF-PEKS scheme, including the definitions of bilinear maps, and the underlying assumptions. Section "Secure-channel free public key encryption with keyword search (SCF-PEKS)" reviews the definition and security model of SCF-PEKS. Section "Proposed efficient SCF-PEKS scheme" presents our proposed SCF-PEKS scheme and gives the security proof in the standard model. Section "Application of SCF-PEKS for EMRs in medical cloud storage" describes the application of SCF-PEKS for EMRs in medical cloud storage. We conclude the paper in Section "Conclusion".

Preliminaries

We first present some notations used throughout this paper. For a prime p, let $\mathbb{Z}_p$ denote the set $\{0, 1, \cdots, p-1\}$, and $\mathbb{Z}_p^*$ denote $\mathbb{Z}_p \backslash \{0\}$. For a finite set S, $x \in_R S$ means choosing an element x from S with a uniform distribution.

Bilinear forms

We write $\mathbb{G}_1 = \langle g \rangle$ to denote that g generates the group $\mathbb{G}_1$. Let $\texttt{GlobalSetup}$ be an algorithm that, on input the security parameters k, outputs the parameters for a bilinear map as $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$, where $\mathbb{G}_1, \mathbb{G}_2$ have prime order p and $\langle g \rangle = \mathbb{G}_1$. The efficient mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ is bilinear for all $g \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$; and non-degenerate, if g generates $\mathbb{G}_1$, then $e(g, g) \neq 1$.

Complexity assumption

Definition 1 (**Quotient Decisional Bilinear Diffie-Hellman (QDBDH)** [2]) Let $\texttt{GlobalSetup}(1^k) \rightarrow (p, g, \mathbb{G}_1, \mathbb{G}_2, e)$, where $\langle g \rangle = \mathbb{G}_1$. For all PPT adversaries $\mathcal{A}$, the following probability is strictly less than $1/2 + 1/poly(k)$ where $poly(k)$ represents any polynomial function in k:

$$|\texttt{Pr}[\mathcal{A}(g, g^a, g^b, e(g, g)^{b/a}) = 1 | a, b \in_R \mathbb{Z}_p^*] \\ - \texttt{Pr}[\mathcal{A}(g, g^a, g^b, e(g, g)^z) = 1 | a, b, z \in_R \mathbb{Z}_p^*]|.$$

Definition 2 (**Decisional Bilinear Diffie-Hellman (DBDH) Assumption**) Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ be a bilinear map. We define the advantage function

$$\texttt{Adv}_{\mathbb{G}_1, \mathcal{A}}^{DBDH}(1^k)$$

of an adversary $\mathcal{A}$ as

$$|\texttt{Pr}[\mathcal{A}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 1] \\ - \texttt{Pr}[\mathcal{A}(g, g^a, g^b, g^c, Q) = 1]|,$$

where $a, b, c \in \mathbb{Z}_p^*$, $Q \in \mathbb{G}_2^*$ are randomly chosen. We say that the decisional bilinear Diffie Hellman assumption holds if $Adv_{\mathbb{G}_1, \mathcal{A}}^{DBDH}(1^k)$ is negligible for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$.

Definition 3 (**Hash Diffie-Hellman (HDH) Problem** [1]) Let $\texttt{hLen}$ be in $\mathbb{N}$ and $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\texttt{hLen}}$ be a hash function. The HDH problem in $\mathbb{G}$ is defined as follows: given $(g, g^a, g^b, H(g^c)) \in \mathbb{G}^3 \times \{0, 1\}^{\texttt{hLen}}$ and $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\texttt{hLen}}$ as inputs, outputs "yes" if $a \cdot b = c$ and "no" otherwise. An algorithm $\mathcal{A}$ that outputs $b' \in \{0, 1\}$ has an advantage ϵ in solving the HDH problem in $\mathbb{G}$ if

$$|\texttt{Pr}[\mathcal{A}(g, g^a, g^b, H(g^{ab})) = \text{“}yes\text{”} : g \leftarrow \mathbb{G}, \\ a, b \leftarrow \mathbb{Z}_p] - \texttt{Pr}\Big[\mathcal{A}(g, g^a, g^b, \eta) = \text{“}yes\text{”} : \\ g \leftarrow \mathbb{G}, \eta \leftarrow \{0, 1\}^{\texttt{hLen}}, a, b \leftarrow \mathbb{Z}_p\Big]| \geq \epsilon,$$

where the probability is taken over the random choice of $g \in \mathbb{G}$, the random choice of $\eta \leftarrow \{0, 1\}^{\texttt{hLen}}$, the random bits of $\mathcal{A}$. We say that the HDH assumption holds in $\mathbb{G}$ if no t-time algorithm has an advantage at least ϵ in solving the HDH problem in $\mathbb{G}$.

Secure-channel free public key encryption with keyword search (SCF-PEKS)

In this section, we review the definition and security model of SCF-PEKS as defined in [20].

Definition of SCF-PEKS

Definition 4 (**SCF-PEKS**) A secure-channel free public key encryption with keyword search scheme consists of the following algorithms:

- $\texttt{GlobalSetup}(1^k)$: The algorithm inputs a security parameter k and outputs the global parameters *params*, which includes a description of keyword space $\mathcal{KS}$.
- $\texttt{KeyGen}_\texttt{R}(params)$: Given the global parameters *params*, the key generation algorithm $\texttt{KeyGen}_\texttt{R}$ outputs a public/private key pair (pk_R, sk_R) of a receiver R.
- $\texttt{KeyGen}_\texttt{S}(params)$: Given the global parameters *params*, the key generation algorithm $\texttt{KeyGen}_\texttt{S}$ outputs a public/private key pair (pk_s, sk_s) of a server S.
- $\texttt{PEKS}(params, pk_R, pk_s, w)$: On input the global parameters *params*, a receiver R's public key pk_R, a server's public key pk_s, a keyword $w \in \mathcal{KS}$, outputs a keyword ciphertext CT of w.
- $\texttt{dTrapdoor}(params, sk_R, pk_s, w)$: Given the global parameters *params*, a receiver R's private key sk_R, a server's public key pk_s and a keyword $w \in \mathcal{KS}$, the trapdoor generation algorithm $\texttt{dTrapdoor}$ outputs a trapdoor T_w of the keyword w corresponding to the receiver R. This algorithm is performed by the private key's owner, who will send the trapdoor to the server. Our scheme does not use secure channel when the user transmits the trapdoor to the server.
- $\texttt{dTest}(params, CT, sk_s, pk_R, T_w)$: Given the global parameters *params*, a server's private key sk_s, a receiver's public key pk_R, a trapdoor T_w, and a PEKS ciphertext $CT = \texttt{PEKS}(params, pk_s, pk_R, w')$, the test algorithm $\texttt{dTest}$ outputs "yes" if $w = w'$ or "no" otherwise.

Correctness For all (pk_R, sk_R) output by $\texttt{KeyGen}_\texttt{R}$ and (pk_s, sk_s) output by $\texttt{KeyGen}_\texttt{S}$, the following equation holds for a correctly generated SCF-PEKS ciphertext associated with keyword w:

$$\texttt{dTest}\,(\texttt{PEKS}(params, pk_R, pk_s, w), sk_s, pk_R, \\ \texttt{dTrapdoor}(params, sk_R, pk_s, w)) = yes$$

Consistency Suppose there exists an adversary $\mathcal{A}$ that wants to make consistency fail. The consistency is formally defined as follows [19]:

```
Experiment $Exp_{\mathcal{A}}^{cons}(1^k)$
$(pk_R, sk_R) \leftarrow \texttt{KeyGen}_\texttt{R}(1^k)$; $(pk_s, sk_s) \leftarrow \texttt{KeyGen}_\texttt{S}(1^k)$;
$(w, w') \leftarrow \mathcal{A}(pk_R, pk_s)$;
$CT \leftarrow \texttt{PEKS}(pk_s, pk_R, w)$;
$T_{w'} \leftarrow \texttt{dTrapdoor}(pk_s, sk_R, w')$;
if$w \neq w'$ and $\texttt{dTest}(T_{w'}, sk_s, pk_R, CT) =$ “yes”.
    then return 1,
    else return 0.
```

We define the advantage of $\mathcal{A}$ as:

$$\texttt{Adv}_{\mathcal{A}}^{cons}(1^k) = \texttt{Pr}[Exp_{\mathcal{A}}^{cons}(1^k) = 1].$$

The scheme is said to be computationally consistent if it is negligible for polynomial time adversaries $\mathcal{A}$ to win the above experiment.

Security model of SCF-PEKS

In an SCF-PEKS scheme, we consider two types of adversary, namely, a malicious server and a malicious user. A malicious server should not be able to distinguish which keyword corresponds to a given keyword ciphertext without the trapdoor from a receiver. A malicious user (including the receiver) should not be able to distinguish which keyword corresponds to a target ciphertext without the server's private key even s/he has the trapdoor of the keyword. We review the security model as defined in [20].

Definition 5 (IND-SCF-CKCA game) Let k be the security parameter and $\mathcal{A}_i$ $(i = 1, 2)$ be the adversary. We consider the following two games between the adversary $\mathcal{A}_i$ $(i = 1, 2)$ and the challenger $\mathcal{B}$.

Game$_{\textbf{Server}}$: $\mathcal{A}_1$ is assumed to be a malicious server.

- `Setup.` $\mathcal{B}$ generates public parameters and gives $\mathcal{A}_1$. $\mathcal{B}$ also runs $\texttt{KeyGen}_\texttt{R}(params) \rightarrow (pk_R, sk_R)$, $\texttt{KeyGen}_\texttt{S}(params) \rightarrow (pk_s, sk_s)$ and returns pk_R and (pk_s, sk_s) to $\mathcal{A}_1$.
- `Query Phase 1.` $\mathcal{A}_1$ makes the following queries.
  - `dTrapdoor` oracle $\mathcal{O}_{T_w}(w)$: $\mathcal{A}_1$ can adaptively ask $\mathcal{B}$ for the trapdoor T_w for any keyword w of his choice. $\mathcal{B}$ responds the trapdoor $T_w = \texttt{dTrapdoor}(params, sk_R, pk_s, w)$ to $\mathcal{A}_1$.
  - `dTest` oracle $\mathcal{O}_T(CT, w)$: $\mathcal{A}_1$ can adaptively asks $\mathcal{B}$ for the Test query for any keyword w and any PEKS ciphertext $CT$of his choice. $\mathcal{B}$ first makes a trapdoor query on w to get the trapdoor T_w and responds the result of $\texttt{dTest}(params, T_w, pk_s, sk_s, CT)$ to $\mathcal{A}_1$.
- `Challenge.` Once $\mathcal{A}_1$ decides that Query Phase 1 is over, it outputs two keywords (w_0, w_1) from the keyword space that has not been queried in Phase 1. $\mathcal{B}$ randomly chooses a bit $\delta \in \{0, 1\}$ and returns the challenge PEKS ciphertext $CT^* = \texttt{PEKS}(params, pk_R, pk_s, w_\delta)$ to $\mathcal{A}_1$.
- `Query Phase 2.` $\mathcal{A}_1$ issues a number of queries from $\mathcal{O}_{T_w}$ and $\mathcal{O}_T$ as in Phase 1. The restriction here is that w_0 and w_1 are not allowed to be queried from $\mathcal{O}_{T_w}$ and $\langle CT, w\rangle$ is not allowed to be queried from $\mathcal{O}_T$ if $\langle CT, w\rangle = \langle CT^*, w_0\rangle$ or $\langle CT, w\rangle = \langle CT^*, w_1\rangle$.
- `Guess.` $\mathcal{A}_1$ outputs the guess δ'. The adversary wins if $\delta' = \delta$.

We define $\mathcal{A}_1$'s advantage in $Game_{Server}$ by

$$\texttt{Adv}_{\mathcal{A}_1}^{Game_{Server}}(k) = |\texttt{Pr}[\delta = \delta'] - 1/2|.$$

Game$_{\textbf{Receiver}}$: $\mathcal{A}_2$ is assumed to be an outsider adversary (including a malicious receiver).

- `Setup.` $\mathcal{B}$ generates the server's public and private key pair (pk_s, sk_s) and the receiver's public and private key pair (pk_R, sk_R) and gives pk_s, (pk_R, sk_R) to $\mathcal{A}_2$.
- `Query Phase 1.` $\mathcal{A}_2$ makes the following query:
  - `dTest` oracle $\mathcal{O}_T(CT, w)$: On input (CT, w) by $\mathcal{A}_2$, $\mathcal{B}$ first makes a trapdoor query on w to get trapdoor T_w and responds the result of $\texttt{dTest}(params, T_w, pk_s, sk_s, CT)$ to $\mathcal{A}_2$.
- `Challenge.` $\mathcal{A}_2$ outputs a target keyword pair (w_0, w_1) from the keyword space. $\mathcal{B}$ randomly chooses a bit $\delta \in \{0, 1\}$ and returns the challenge PEKS ciphertext $CT^* = \texttt{PEKS}(params, pk_R, pk_s, w_\delta)$ to $\mathcal{A}_2$.
- `Query Phase 2.` $\mathcal{A}_2$ issues a number of queries from $\mathcal{O}_T$ as in Phase 1. The restriction here is that $\langle CT, w\rangle$ is not allowed to be queried from $\mathcal{O}_T$ if $\langle CT, w\rangle = \langle CT^*, w_0\rangle$ or $\langle CT, w\rangle = \langle CT^*, w_1\rangle$.
- `Guess.` $\mathcal{A}_2$ outputs the guess δ'. The adversary wins if $\delta' = \delta$.

We define $\mathcal{A}_2$'s advantage in $Game_{Receiver}$ by

$$\texttt{Adv}_{\mathcal{A}_2}^{Game_{Receiver}}(k) = |\texttt{Pr}[\delta = \delta'] - 1/2|.$$

The SCF-PEKS scheme is said to be IND-SCF-CKCA secure if $Adv_{\mathcal{A}_i}^{Game_j}(k)$, is negligible, where $((i = 1 \wedge j = Server) \vee (i = 1 \wedge j = Receiver))$.

SCF-PEKS secure against off-line keyword guessing attacks

Rhee et al. introduced the concept of trapdoor indistinguishability and showed that if an SCF-PEKS scheme satisfies trapdoor indistinguishability, the scheme can resist off-line keyword-guessing attacks [28]. Fang et al. [20] also proposed a similar notion of indistinguishability of SCF-PEKS against keyword guessing attacks (IND-KGA). In this subsection, we review the security against off-line keyword guessing attacks of SCF-PEKS as defined in [20].

Definition 6 (**IND-KGA game**) Let $\mathcal{A}_3$ be an outsider adversary (neither the server nor the receiver) that performs the off-line keyword guessing attack. Let k be the security parameter, the security game is defined as follows:

- $\texttt{Setup.}$ The global parameter generation algorithm, $\texttt{GlobalSetup}(1^k)$, the two key generation algorithms, $\texttt{KeyGen}_\texttt{R}(params)$ and $\texttt{KeyGen}_\texttt{S}(params)$, are run. $params$, pk_R, pk_s are given to $\mathcal{A}_3$ while sk_R and sk_s are kept secret from $\mathcal{A}_3$.
- $\texttt{Query Phase 1.}$ $\mathcal{A}_3$ makes the following query:
    - $\texttt{dTrapdoor}$ oracle $\mathcal{O}_{T_w}(w)$: $\mathcal{A}_3$ can adaptively ask $\mathcal{B}$ for the trapdoor T_w of any keyword w of his choice. $\mathcal{B}$ responds the trapdoor $T_w = \texttt{dTrapdoor}(params, sk_R, pk_s, w)$ to $\mathcal{A}_3$.
- $\texttt{Challenge.}$ $\mathcal{A}_3$ gives $\mathcal{B}$ two keywords w_0 and w_1, on which it wishes to be challenged. The restriction is that the corresponding trapdoors T_{w_0} and T_{w_1} have not been queried by the adversary in Phase 1. $\mathcal{B}$ picks a random $\delta \in \{0, 1\}$ and returns the trapdoor $T_{w_\delta} = \texttt{dTrapdoor}(params, pk_R, pk_s, w_\delta)$ to $\mathcal{A}_3$.
- $\texttt{Query Phase 2.}$ $\mathcal{A}_3$ issues a number of queries from $\mathcal{O}_{T_w}$ as in Phase 1. The restriction here is that w_0 and w_1 are not allowed to be queried from $\mathcal{O}_{T_w}$.
- $\texttt{Guess.}$ $\mathcal{A}_3$ outputs the guess $\delta' \in \{0, 1\}$ and wins in the IND-KGA game, if $\delta' = \delta$.

We define $\mathcal{A}_3$'s advantage in the IND-KGA game by

$$\texttt{Adv}_{\mathcal{A}_3}^{IND-KGA}(1^k) = |\texttt{Pr}[\delta = \delta'] - 1/2|.$$

The SCF-PEKS scheme is said to be IND-KGA attack secure if $Adv_{\mathcal{A}_3}^{IND-KGA}(k)$ is negligible.

Proposed efficient SCF-PEKS scheme

Our construction

The description of our SCF-PEKS scheme is as follows.

- $\texttt{GlobalSetup}(1^k)$: Let k be the security parameter and $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$ be the bilinear map parameters. g_1, u, v, d, h are random generators in $\mathbb{G}_1$. Select collision-resistant hash functions $H : \mathbb{G}_1 \rightarrow \mathbb{G}_1$, $H_1 : \mathbb{G}_2 \longrightarrow \{0, 1\}^k$ and $H_2 : \mathbb{G}_1 \times \{0, 1\}^k \longrightarrow \mathbb{Z}_p^*$. Output the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, e, g, g_1, u, v, d, h, H, H_1, H_2, \mathcal{KS})$, where $\mathcal{KS}$ is a description of keyword space.
- $\texttt{KeyGen}_\texttt{R}(params)$: On input $params$, a receiver R selects a random $x_R \in \mathbb{Z}_p^*$ and sets the public key as $pk_R = g^{x_R}$ and private key as $sk_R = x_R$.
- $\texttt{KeyGen}_\texttt{S}(params)$: A server S selects a random $x_s \in \mathbb{Z}_p^*$ and sets the public key as $pk_s = g^{x_s}$ and private key as $sk_s = x_s$.
- $\texttt{PEKS}(params, pk_R, pk_s, w)$: On input a receiver R's public key pk_R, a server S's public key pk_s and a keyword w, a sender computes the keyword ciphertext as follows:
    1. Pick $r \in_R \mathbb{Z}_p^*$ and $C_1 = pk_R^r$. Compute $T = e(pk_s, g_1^w h)^r$, $C_2 = H_1(T)$.
    2. Pick $s \in_R \mathbb{Z}_p^*$, compute $h' = H_2(C_1, C_2)$ and $C_3 = (u^{h'} v^s d)^r$.
    3. Output the PEKS ciphertext $CT = (s, C_1, C_2, C_3)$.
- $\texttt{dTrapdoor}(params, sk_R, pk_s, w)$: On input a receiver R's private key sk_R, a server's public key pk_s and a keyword w, the receiver R randomly selects $r' \in_R \mathbb{Z}_p^*$, computes $T_1 = g^{r'}$ and $T_2 = (g_1^w h)^{1/sk_R} \cdot H(pk_s^{r'})$, outputs the trapdoor associated with the keyword w as $T_w = (T_1, T_2)$.
- $\texttt{dTest}(params, CT, sk_s, pk_R, T_w)$: On input a server's private key $sk_s = x_s$, a trapdoor $T_w = (T_1, T_2)$ and a ciphertext $CT = (s, C_1, C_2, C_3)$, the server perform the following computation:

    Compute $h' = H_2(C_1, C_2)$ and test if equation

    $$e(C_1, (u^{h'} v^s d)) = e(pk_R, C_3) \tag{1}$$

    is valid. If not, output $\perp$. Otherwise, the server computes $\mathcal{T} = T_2/H(T_1^{x_s})$, and checks if $C_2 = H_1(e(C_1, \mathcal{T}^{x_s}))$. If the equality is satisfied, then output "yes"; otherwise, output "no".

Correctness We show that a correctly generated PEKS ciphertext can be correctly tested by the server who has the correct trapdoor. Let a PEKS ciphertext $CT = (s, C_1, C_2, C_3)$ associated with keyword w under the public key pk_s and pk_R. Let the trapdoor $T_w = (T_1, T_2)$, where $T_1 = g^{r'}$ and $T_2 = (g_1^w h)^{1/x_R} \cdot H(pk_s^{r'})$. We have

$$\mathcal{T} = T_2/H(T_1^{x_s}) = \frac{(g_1^w h)^{1/x_R} \cdot H(pk_s^{r'})}{H(pk_s^{r'})} = (g_1^w h)^{\frac{1}{x_R}},$$

$$\begin{aligned} H_1(e(C_1, \mathcal{T}^{x_s})) &= H_1(e(pk_R^r, (g_1^w h)^{\frac{x_s}{x_R}})) \\ &= H_1(e(pk_s, g_1^w h)^r) \\ &= C_2. \end{aligned}$$

Therefore, we have $\mathtt{dTest}(params, \mathtt{PEKS}(params, pk_R, pk_s, w), sk_S, pk_R, \mathtt{dTrapdoor}(params, sk_R, pk_s, w)) = yes$.

Consistency Let $r, r' \in_R \mathbb{Z}_p^*$ denote two values chosen randomly by the SCF-PEKS scheme. Let $C_1 = pk_R^r$, $C_2 = H_1(e(pk_s, g_1^w h)^r)$ be the partial ciphertext associated with the keyword w. Let $T_{w'} = (T_1', T_2')$ be the trapdoor associated with the keyword w', where $T_1' = g^{r'}$ and $T_2' = (g_1^{w'} h)^{1/x_R} \cdot H(pk_s^{r'})$.

If $H_1(e(C_1, (T_2'/H(T_1^{'x_s}))^{x_s})) = C_2$

$$\begin{aligned} &\Leftrightarrow H_1(e(pk_R^r, (\frac{((g_1^{w'} h)^{1/x_R} \cdot H(pk_s^{r'}))}{H(pk_s^{r'})})^{x_s})) = H_1(e(pk_s, g_1^w h)^r) \\ &\Leftrightarrow H_1(e(pk_R^r, (g_1^{w'} h)^{\frac{x_s}{x_R}})) = H_1(e(pk_s, g_1^w h)^r) \\ &\Leftrightarrow H_1(e(pk_s, g_1^{w'} h)^r) = H_1(e(pk_s, g_1^w h)^r). \end{aligned}$$

But $w \neq w'$, and H_1 is a collision-resistant hash function. Therefore, it holds $H_1(e(C_1, (T_2'/H(T_1^{'x_s}))^{x_s}) \neq C_2$ with a high probability.

Security of our SCF-PEKS scheme

In this subsection, we analyze the security of our SCF-PEKS scheme in the standard model. The analysis of $Game_{Server}$ and $Game_{Receiver}$ as follows.

Theorem 1 *The above scheme is IND-SCF-CKCA secure in the standard model assuming that QDBDH problem and DBDH problem are intractable.*

Lemma 1 *Our scheme is semantically secure against a chosen keyword and ciphertext attacks in $Game_{Server}$ in the standard model assuming QDBDH problem is intractable.*

Proof We assume that $\mathcal{A}_1$ is a malicious server with an advantage ϵ in breaking the proposed scheme. We assume that H, H_1 and H_2 are target collision resistant. Then suppose that there exists an adversary $\mathcal{A}_1$ who can break the (q_{T_w}, q_T, ϵ)-IND-SCF-CKCA security of our SCF-PEKS scheme, where q_{T_w} denotes the times of trapdoor queries and q_T denotes the times of test queries. We can construct an algorithm $\mathcal{B}$ which can break the QDBDH assumption with ϵ' in $(\mathbb{G}_1, \mathbb{G}_2)$ with $\epsilon' \geq \frac{\epsilon}{e \cdot q_{T_w}} - Adv_H^{TCR}$.

Suppose algorithm $\mathcal{B}$ is given a QDBDH instance $(g, A = g^a, B = g^b, Q) \in (\mathbb{G}_1)^3 \times \mathbb{G}_2$ with unknown $a, b \in_R \mathbb{Z}_p^*$. $\mathcal{B}$'s goal is to decide whether $Q = e(g, g)^{b/a}$. In the $Game_{Server}$ $\mathcal{B}$ works by interacting with adversary $\mathcal{A}_1$ as follows:

- `Setup.` $\mathcal{B}$ chooses random $x_u, x_v \in \mathbb{Z}_p$ and sets $g_1 = B^{\alpha_0}$, $h = B^{\beta}$, $u = (g^{x_u} A^{\alpha_1})$, $v = (g^{x_v} A^{\alpha_2})$, and $d = A^{\alpha_3}$ for random $\alpha_0, \beta, \alpha_1, \alpha_2, \alpha_3 \in_R \mathbb{Z}_p^*$ and provides them to $\mathcal{A}_1$. $\mathcal{B}$ picks $x_R \in \mathbb{Z}_p^*$. Next, using the Corons technique [8], it flips a biased coin $c_i \in \{0, 1\}$ that yields 1 with probability θ and 0 otherwise. If $c_i = 1$, it sets $pk_R = g^{x_R}$; else $pk_R = A^{x_R}$. Next, $\mathcal{B}$ adds the tuple (pk_R, x_R, c_i) to L^{List}. $\mathcal{B}$ generates the server's public and private key pair (pk_s, sk_s), and returns pk_R and (pk_s, sk_s) to $\mathcal{A}_1$.
- `Query Phase 1.` $\mathcal{A}_1$ issues a series of queries . $\mathcal{B}$ maintains a list L^{list} and answers these queries for $\mathcal{A}_1$ as follows:
    - `dTrapdoor` oracle $\mathcal{O}_{T_w}(w)$: $\mathcal{B}$ randomly selects $r' \in \mathbb{Z}_p^*$. If $c_i = 1$, it means that $sk_R = x_R$, $\mathcal{B}$ outputs $T_w = (T_1, T_2)$, where $T_1 = g^{r'}$, $T_2 = (g_1^w h)^{1/x_R} \cdot H(pk_s^{r'})$. If $c_i = 0$, $\mathcal{B}$ outputs a random bit in $\{0, 1\}$ and aborts.
    - `dTest` oracle $\mathcal{O}_T(CT, w)$: $\mathcal{A}_1$ asks $\mathcal{B}$ for the test query of keyword w and PEKS ciphertext CT of his choice. $\mathcal{B}$ computes $h' = H_2(C_1, C_2)$ and then tests if $e(C_1, u^{h'} v^s d) = e(pk_R, C_3)$ holds. Then $\mathcal{B}$ first query a trapdoor query on $\langle w \rangle$. If $c_i = 1$, $\mathcal{B}$ gets the trapdoor T_w and then responds by sending the result of $\mathtt{dTest}(params, T_w, sk_s, pk_R, CT)$ to $\mathcal{A}_1$. If $c_i = 0$, $pk_R = A^{x_R}$. $\mathcal{B}$ have $C_1 = pk_R^r = A^{x_R \cdot r}$, $C_3 = (u^{h'} v^s d)^r = (g^{x_u h' + x_v s} A^{\alpha_1 h' + \alpha_2 s + \alpha_3})^r$. $\mathcal{B}$ can deduce $g^r = (\frac{C_3}{C_1^{\frac{\alpha_1 h' + \alpha_2 s + \alpha_3}{x_R}}})^{\frac{1}{x_u h' + x_v s}}$. $\mathcal{B}$ checks if

      $$H_1(e(g^r, g_1^w h)^{x_s}) = C_2 \tag{2}$$

      is valid, because

      $$H_1(e(g^r, g_1^w h)^{x_s}) = H_1(e(pk_s, g_1^w h)^r) = C_2.$$

      Then $\mathcal{B}$ responds by sending the result of $\mathtt{dTest}(params, T_w, sk_s, pk_R, CT)$ to $\mathcal{A}_1$.

- `Challenge.` When Phase 1 is over, $\mathcal{A}_1$ outputs a challenge tuple (pk_{R^*}, w_0, w_1). $\mathcal{B}$ responds as follows:

1. Recover tuple $(pk_{R^*}, x_{R^*}, c_{i^*})$ from L^{list}. If $c_{i^*} = 1$, $\mathcal{B}$ outputs a random bit in $\{0, 1\}$ and aborts. Otherwise, it means that $pk_{R^*} = A^{x_{R^*}}$ and $\mathcal{B}$ proceeds to execute the rest of the steps.
2. Pick $\delta \in_R \{0, 1\}$, define $C_1^* = g^{x_{R^*}}$ and compute $T^* = Q^{x_s(w_{\delta\alpha_0}+\beta)}$, $C_2^* = H_1(T^*)$. It sets $h'^* = H_2(C_1^*, C_2^*)$, $s^* = \frac{-x_u h'^*}{x_v}$ and compute $C_3^* = g^{\alpha_1 h'^* + \alpha_2 s^* + \alpha_3}$. Finaly, $\mathcal{B}$ returns $CT^* = (s^*, C_1^*, C_2^*, C_3^*)$ as the challenge ciphertext to $\mathcal{A}_1$.

   If $Q = e(g, g)^{\frac{b}{a}}$, CT^* is indeed a valid challenge PEKS ciphertext under public key pk_{R^*}. To see this, let $r^* = \frac{1}{a}$, we have

$$C_1^* = g^{x_{R^*}} = (g^a)^{x_{R^*} \cdot \frac{1}{a}} = (A^{x_{R^*}})^{r^*} = pk_{R^*}^{r^*},$$
$$C_2^* = H_1(Q^{x_s(w_{\delta\alpha_0}+\beta)}) = H_1(e(pk_s, g^{(w_{\delta\alpha_0}+\beta)})^{\frac{b}{a}})$$
$$= H_1(e(pk_s, B^{(\alpha_0 w_\delta + \beta)})^{r^*}) = H_1(e(pk_s, (g_1^{w_\delta} h))^{r^*}),$$
$$C_3^* = g^{(\alpha_1 h'^* + \alpha_2 s^* + \alpha_3)} = (A^{\alpha_1 h'^* + \alpha_2 s^* + \alpha_3})^{r^*} =$$
$$(g^{x_u h'^* + x_v s^*} \cdot A^{\alpha_1 h'^*} \cdot A^{\alpha_2 s^*} \cdot A^{\alpha_3})^{r^*} = ((g^{x_u} A^{\alpha_1})^{h'^*} \cdot$$
$$(g^{x_v} A^{\alpha_2})^{s^*} \cdot A^{\alpha_3})^{r^*} = (u^{h'^*} v^{s^*} d)^{r^*}.$$

On the other hand, when Q is uniform and independent in $\mathbb{G}_2$, the challenge ciphertext ciphertext CT^* is independent of δ in the adversary's view.

- `Query Phase 2.` $\mathcal{A}_1$ continues making queries as in the Query Phase 1. The restriction is that w_0 and w_1 are not allowed to be queried from $\mathcal{O}_{T_w}$ and $\langle CT, w \rangle$ are not queried from $\mathcal{O}_T$ if $\langle CT, w \rangle = \langle CT^*, w_0 \rangle$ or $\langle CT, w \rangle = \langle CT^*, w_1 \rangle$. otherwise, $\mathcal{B}$ returns $\perp$ which is not in the trapdoor space.
- `Guess.` Eventually, $\mathcal{A}_1$ returns a guess $\delta' \in \{0, 1\}$. If $\delta' = \delta$, $\mathcal{B}$ outputs 1 meaning $Q = e(g, g)^{\frac{b}{a}}$; else, $\mathcal{B}$ outputs 0 meaning $Q = e(g, g)^r$.

Now we begin to analyze the probability. Let Abort denotes the event of $\mathcal{B}$'s aborting during the simulation of oracles $\mathcal{O}_{T_w}$, $\mathcal{O}_T$ or in challenge phase. We have $\mathtt{Pr}[\neg Abort] \geq \theta^{q_{T_w}}(1 - \theta)$ which is maximized at $\theta_{opt} = \frac{q_{T_w}}{1+q_{T_w}}$. Using θ_{opt}, the probability $\mathtt{Pr}[\neg Abort]$ is at least $\frac{1}{e \cdot q_{T_w}}$. Therefore, we have $\epsilon' \geq \frac{\epsilon}{e \cdot q_{T_w}} - Adv_H^{TCR}$, where e denotes the base of the natural algorithm. This completes the proof of lemma 1. □

Lemma 2 *Our scheme is semantically secure against a chosen keyword attack in $Game_{Receiver}$ in the standard model assuming DBDH problem is intractable.*

Proof We assume that $\mathcal{A}_2$ is an outsider adversary (including the receiver) with an advantage ϵ in breaking the proposed scheme. We assume that H, H_1 and H_2 are target collision resistant. Then suppose that there exists an adversary $\mathcal{A}_2$ who can break the ϵ-IND-CKA security of our PEKS scheme. We can construct an algorithm $\mathcal{B}$ which can break the DBDH assumption with $\epsilon' = (\epsilon - Adv_H^{TCR})$ in $(\mathbb{G}_1, \mathbb{G}_2)$.

Suppose algorithm $\mathcal{B}$ is given a DBDH instance $(g, A = g^a, B = g^b, C = g^c, Q) \in (\mathbb{G}_1)^3 \times \mathbb{G}_2$ with unknown $a, b, c \in_R \mathbb{Z}_p^*$. $\mathcal{B}'$s goal is to decide whether $Q = e(g, g)^{abc}$. $\mathcal{B}$ works by interacting with adversary $\mathcal{A}_2$ in the IND-CKA game as follows:

- `Setup.` $\mathcal{B}$ provides $\mathcal{A}_2$ with public parameters $g_1 = B = g^b$, $h = B^\beta$, $u = A^{\alpha_1}$, $v = A^{\alpha_2}$, and $d = g^{\alpha_3}$ for random $\alpha_1, \alpha_2, \alpha_3, \beta \in_R \mathbb{Z}_p^*$. Let $pk_s = A = g^a$ be the server's public key. $\mathcal{B}$ randomly chooses $x_R \in \mathbb{Z}_p^*$ and sets $pk_R = g^{x_R}$ and $sk_R = x_R$ as the receiver's public and private key respectively. $\mathcal{B}$ sends (pk_R, sk_R) to $\mathcal{A}_2$.
- `Query Phase 1.` $\mathcal{A}_2$ queries the `dTest` oracle as follows:
  - `dTest` oracle $\mathcal{O}_T(CT, w)$: $\mathcal{A}_2$ can adaptively ask $\mathcal{B}$ for the test query of any keyword w and any PEKS ciphertext $CT = (s, C_1, C_2, C_3)$ of his choice. $\mathcal{B}$ computes $h' = H_2(C_1, C_2)$ and then tests if $e(C_1, u^{h'} v^s d) = e(pk_R, C_3)$ holds. Since $C_1 = pk_R^r = (g^r)^{x_R}$, $C_3 = (u^{h'} v^s d)^r = (A^{\alpha_1 h' + \alpha_2 s} g^{\alpha_3})^r$, we have $g^r = C_1^{\frac{1}{x_R}}$ and $A^r = (\frac{C_3}{C_1^{\frac{\alpha_3}{x_R}}})^{\frac{1}{\alpha_1 h' + \alpha_2 s}}$. $\mathcal{B}$ checks if

    $$H_1(e(g_1^w h, A^r)) = C_2,$$

    because $H_1(e(g_1^w h, A^r)) = H_1(e(g_1^w h, pk_s)^r) = C_2$. $\mathcal{B}$ then randomly selects $r' \in \mathbb{Z}_p^*$, and computes $T_w = (T_1, T_2)$, where $T_1 = g^{r'}$, $T_2 = (g_1^w h)^{1/x_R} \cdot H(pk_s^{r'})$. $\mathcal{B}$ returns the result of $\mathtt{dTest}(params, T_w, sk_s, pk_R, CT)$ to $\mathcal{A}_2$.
- `Challenge.` When Phase 1 is over, $\mathcal{A}_2$ outputs a challenge tuple (pk_{R^*}, w_0, w_1). $\mathcal{B}$ responds by choosing a random $\delta \in \{0, 1\}$. Let the challenge keyword be $w^* = w_\delta$, $\mathcal{B}$ computes $C_1^* = (g^c)^{x_R^*}$, $T^* = H_1(Q^{w^*+\beta})$, $h'^* = H_2(C_1^*, C_2^*)$, $s^* = \frac{-\alpha_1 h'^*}{\alpha_2}$, and $C_3^* = C^{\alpha_3}$. $\mathcal{B}$ sends the challenge PEKS ciphertext $C^* = (s^*, C_1^*, C_2^*, C_3^*)$ to $\mathcal{A}_2$.

  If $Q = e(g, g)^{abc}$, CT^* is indeed a valid challenge ciphertext under public key pk_{R^*}. To see this, let $r^* = c$, we have

$$C_1^* = (g^c)^{x_{R^*}} == (g^{x_{R^*}})^{r^*} = pk_{R^*}^{r^*},$$
$$C_2^* = H_1(Q^{(w^*+\beta)}) = H_1(e(g, g)^{abc(w^*+\beta)})$$
$$= H_1(e(g^a, g^{b(w^*+\beta)})^c) = H_1(e(pk_s, g_1^{w^*} h)^{r^*}),$$
$$C_3^* = (g^c)^{\alpha_3} = (g^{\alpha_3})^c = (A^{\alpha_1 h'^*} \cdot A^{\alpha_2(\frac{-\alpha_1 h'^*}{\alpha_2})} \cdot$$
$$g^{\alpha_3})^c = (A^{\alpha_1 h'^*} \cdot A^{\alpha_2 s^*} \cdot g^{\alpha_3})^c = (u^{h'^*} v^{s^*} d)^{r^*}.$$

- `Query Phase 2.` $\mathcal{A}_2$ issues a number of queries from $\mathcal{O}_T$ as in Phase 1. The restriction here is that $\langle CT, w\rangle$ is not allowed to be queried from $\mathcal{O}_T$ if $\langle CT, w\rangle = \langle CT^*, w_0\rangle$ or $\langle CT, w\rangle = \langle CT^*, w_1\rangle$. Otherwise, $\mathcal{B}$ returns $\perp$.
- `Guess.` Eventually, $\mathcal{A}_2$ returns a guess $\delta' \in \{0, 1\}$. If $\delta' = \delta$, $\mathcal{B}$ outputs 1 meaning $Q = e(g, g)^{abc}$; else, $\mathcal{B}$ outputs 0 meaning $Q = e(g, g)^r$.

□

Off-line keyword guessing attack resiliency

Theorem 2 *Our SCF-PEKS scheme is IND-KGA secure in the standard model, under the assumption that Hash Diffie-Hellman (HDH) is intractable.*

Proof Let $\mathcal{A}_3$ be an outsider adversary who makes at most q_{T_w} trapdoor queries. Assume that $\mathcal{A}_3$ has an advantage ϵ in breaking IND-KGA Game of the proposed scheme, we build an algorithm $\mathcal{B}$ which has an advantage $\epsilon' = \epsilon$ in solving the HDH problem in $\mathbb{G}_1$. $\mathcal{B}$ takes as input a random HDH instance $(g, A = g^a, B = g^b, \eta) \in \mathbb{G}_1$, and $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$, where H is a hash function and η is either $H(g^{ab})$ or a random element of $\mathbb{G}_1$.

- `Setup.` Algorithm $\mathcal{B}$ randomly chooses $g_1 \in \mathbb{G}_1$, $x_R \in \mathbb{Z}_p^*$ and sets the receiver R's private key $sk_R = x_R$ and public key $pk_R = g^{x_R}$. It chooses a random value $l \in_R \mathbb{Z}_p^*$ and sets the server's public key $pk_s = A^l = (g^a)^l$, where the private key of the server is implicitly defined as $sk_s = al$. $\mathcal{B}$ sends (pk_R, pk_s) to $\mathcal{A}_3$.
- `Query Phase 1.`
    - `dTrapdoor` oracle $\mathcal{O}_{T_w}(w)$: When $\mathcal{A}_3$ issues a query for a trapdoor that corresponds to the keyword w_j, $\mathcal{B}$ responds as follows:
        - $\mathcal{B}$ randomly chooses $r' \in \mathbb{Z}_p^*$ and computes $T_1 = g^{r'}$, $T_2 = (g_1^{w_j}h)^{\frac{1}{x_R}} \cdot H(pk_s^{r'})$.
        - $\mathcal{B}$ responds to $\mathcal{A}_3$ with the trapdoor, $T_{w_j} = (T_1, T_2)$ of w_j.
- `Challenge.` $\mathcal{A}_3$ outputs two keywords w_0 and w_1 that she wishes to be challenged on. $\mathcal{B}$ generates the challenge trapdoor $T_{w_\delta} = (T_1^*, T_2^*)$ as follows.
    - $\mathcal{B}$ picks a random bit $\delta \in \{0, 1\}$ and sets $T_1^* = B^{\frac{1}{l}}$, $T_2^* = (g_1^{w_\delta}h)^{\frac{1}{x_R}} \cdot \eta$ where $l \in_R \mathbb{Z}_p^*$ is the value that is selected in the setup phase and η is a component of the HDH challenge.
    - $\mathcal{B}$ responds with the challenge trapdoor $T^*_{w_\delta} = (T_1^*, T_2^*)$.

  If $\eta = H(g^{ab})$, $T^*_{w_\delta} = (T_1^*, T_2^*)$ is a valid trapdoor under public key pk_R. Let $r'^* = \frac{b}{l}$, we have $T_1^* = B^{\frac{1}{l}} = g^{r'^*}$, $T_2^* = (g_1^{w_\delta}h)^{\frac{1}{x_R}} \cdot \eta = (g_1^{w_\delta}h)^{\frac{1}{sk_R}} \cdot H(g^{ab}) = (g_1^{w_\delta}h)^{\frac{1}{sk_R}} \cdot H(g^{al\cdot\frac{b}{l}}) = (g_1^{w_\delta}h)^{\frac{1}{sk_R}} \cdot H(pk_s^{r'^*})$.
- `Query Phase 2.` $\mathcal{A}_3$ can issue trapdoor queries for the keyword w_j. The restriction is that $w_j \neq w_0, w_1$. Algorithm $\mathcal{B}$ responds to these queries as before.
- `Guess.` Eventually, $\mathcal{A}_3$ outputs the guess $\delta' \in \{0, 1\}$, which indicates whether the challenge $T^*_{w_\delta}$ is `dTrapdoor`$(params, sk_R, pk_s, w_0)$ or `dTrapdoor`$(params, sk_R, pk_s, w_1)$. If $\delta = \delta'$, then $\mathcal{B}$ outputs 1, meaning $\eta = H(g^{ab})$; otherwise, it outputs 0, meaning $\eta \in_R \mathbb{G}_1$. □

Performance evaluation

In Table 1, we compare our scheme with Fang et al. [20] (denoted by FS13) and Rhee et al. [28] (denoted by RP10) schemes. We use t_p, t_e, t_s, t_v to represent the computational cost of a bilinear pairing operation, an exponentiation, signing and verifying operations of a one-time signature respectively. "Length of pk" and "Length of sk" denote the length of a public key and a private key, respectively. n denotes the length of keyword space. "Without RO?" denotes whether or not the scheme uses random oracle model in the security proof.

To the best of our knowledge, our SCF-PEKS scheme and the one proposed by Fang et al. in [20] are the only two SCF-PEKS schemes which are proven secure without random oracles. However, our SCF-PEKS scheme only requires a very short key size as compared to Fang et al. scheme [20]. For example, the length of a public key is $|\mathbb{G}_1|$ and the length of a secret key is p in our scheme but the length of a public key is $2|\mathbb{G}_1|+(n+1)|\mathbb{G}_1|$ and a secret key is $2p + (n + 1)p$ in [20]. Here $n = 160$, $p = 2^{160}$, $|\mathbb{G}_1| = 2^{512}$. Therefore, the key size of our proposed SCF-PEKS scheme is 99.4% shorter than the Fang et al.'s scheme [20].

We implemented our proposed SCF-PEKS scheme using JAVA programming language on a Dell Inspiron laptop that operates on Windows 8 (64-bit) with CPU of Intel Core i7-4500U, 1.8GHz and memory of 8GB DDR3L. We run each algorithm for 100 rounds to get an average run time as shown in Table 2. To further show that the SCF-PEKS scheme is feasible to run on a client platform with constrained resources, such as a tablet, we also conducted a preliminary experiment by running the `dTrapdoor` and `PEKS` algorithms on ASUS VivoTab Smart ME400C that operates on Windows 8 with processor of Intel Atom Z2760 Dual-core 1.8GHz and memory of 2GB. The average run

Table 1 Comparisons between FS13, RP10 and our SCF-PEKS scheme

Schemes	FS13 [20]	RP10 [28]	Our scheme
`PEKS`	$3t_p + 6.5t_e + t_s$	$1t_p + 2t_e$	$1t_p + 4t_e$
`dTest`	$4t_p + 3t_e + t_v$	$1t_p + 2t_e$	$3t_p + 3t_e$
Length of pk	$2\lvert\mathbb{G}_1\rvert + (n+1)\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{G}_1\rvert$
Length of sk	$2p + (n+1)p$	$2\lvert\mathbb{G}_1\rvert$	p
IND-CKCA	Yes	No	Yes
IND-KGA	Yes	Yes	Yes
Without RO?	Yes	No	Yes
Test query	Yes	No	Yes

time of `dTrapdoor` and `PEKS` is recorded as 0.38s and 0.48s, respectively.

Application of SCF-PEKS for EMRs in medical cloud storage

With the rapid development of cloud computing and mobile networking technologies, health care practitioners are able to access electronic medical records that stored on a medical cloud storage with mobile devices (e.g., tablets). Confidentiality of the stored contents is one of the major concerns of the patients [25, 26]. The property of confidentiality should also be maintained even if health care practitioners designate the storage provider to search and retrieve patients' records associated with certain keywords.

Consider a medical cloud application that consists of a cloud service provider (CSP) and health care providers that store EMRs on the cloud storage. The health care practitioners encrypt all the stored EMRs to ensure the confidentiality of the contents. To retrieve encrypted EMRs related to a specific keyword by using the conventional approach, the health care practitioners have to download all the stored EMRs, decrypt, and perform their search on local systems. For example, if the medical cloud storage contains 1 gigabyte of EMRs, but only 1 megabyte of data is related to the specific keyword. It is required to retrieve all the 1 gigabyte of data which is inefficient.

Table 2 Average run time (ms) of each algorithm for our SCF-PEKS scheme

Algorithms	Time (ms)
`GlobalSetup`	218.65
$\mathtt{KeyGen_R}$	6.57
$\mathtt{KeyGen_S}$	6.42
`PEKS`	58.10
`dTrapdoor`	51.30
`dTest`	113.70

To solve this problem, we consider the application of SCF-PEKS schemes for many-writer/single-reader (MWSR) setting [22], where a health care practitioner S (sender) who generates and stores the encrypted EMRs is different from another health care practitioner R (receiver) that requests CSP to search and retrieves it from the storage. As shown in Fig. 1, the health care practitioner S who uploads EMRs to the medical cloud storage will first encrypt EMRs with a conventional public key encryption scheme under the public key of the health care practitioner R. In addition, a keyword w associated with the EMRs is encrypted with SCF-PEKS scheme under the public key of both R and the CSP. The health care practitioner R who wants to selectively download certain EMRs that are only related to the keyword w will generate a trapdoor under his/her private key. This trapdoor is then sent to the CSP. Upon receiving this trapdoor, the CSP run the `dTest` algorithm to test the received trapdoor and keyword ciphertexts that stored in the medical cloud storage. It will then return those encrypted EMRs that are associated with w to R.

A complete implementation of such medical cloud application may incorporate other cryptographic primitives and techniques to either enhance its efficiency or functionalities. The complexity of search time for our proposed SCF-PEKS is $O(n)$, where n is the number of encrypted EMRs stored in the medical cloud for a particular medical practitioner. To improve the search efficiency, we may incorporate the hybrid-indexed search method proposed in [33]. The hybrid index consists of a static index and a dynamic index. If a keyword is queried for the fist time, the hybrid-indexed search refers to the static index for searching the encrypted EMRs. While, the dynamic index is used for searching EMRS associated with a keyword that has been queried before. We note that the trapdoor which used as the dynamic index in [33] is deterministic. Therefore, the hybrid-indexed search method in [33] cannot be directly applied to our SCF-PEKS scheme that generates a probabilistic trapdoor.

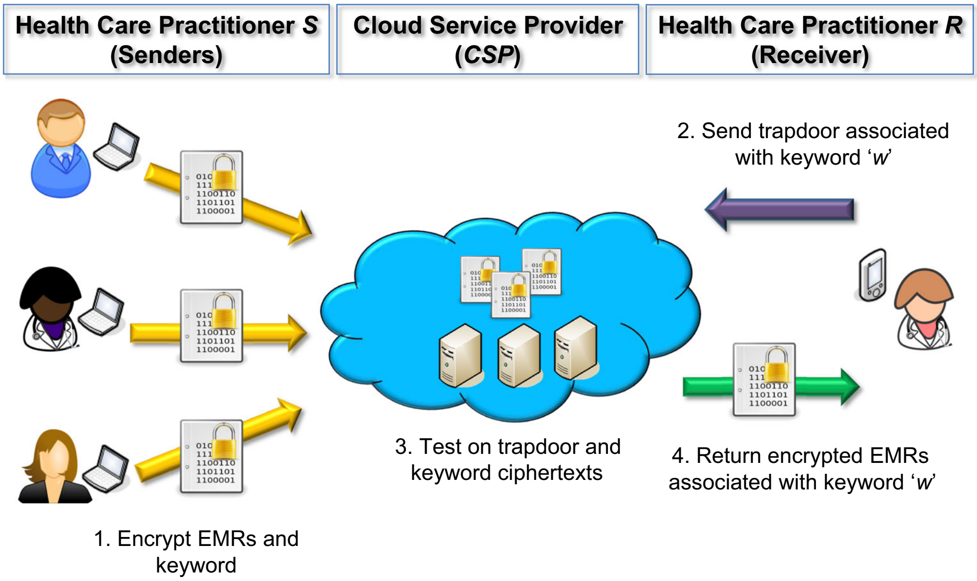
Health Care Practitioner S (Senders)
Cloud Service Provider (CSP)
Health Care Practitioner R (Receiver)
2. Send trapdoor associated with keyword 'w'
3. Test on trapdoor and keyword ciphertexts
4. Return encrypted EMRs associated with keyword 'w'
1. Encrypt EMRs and keyword

Fig. 1 Applying SCF-PEKS for EMRs in medical cloud storage

In addition, we may consider to apply the techniques proposed in [6, 14, 36, 37] to combine the public key encryption (PKE) of the EMRs with our SCF-PEKS. We plan to investigate on how to tweak all these techniques to suit our proposed SCF-PEKS scheme for implementing a real-life application in future.

Conclusion

In this paper, we proposed a very efficient SCF-PEKS scheme that is secure against chosen keyword and ciphertext attacks, and keyword guessing attacks based on the QDBDH, DBDH, and HDH assumptions in the standard model. Our proposed SCF-PEKS scheme is suitable to be used for searching encrypted EMRs in a medical cloud environment which involves mobile devices or client platforms with constrained system resources.

There are several open problems related to this research. First, the construction of both Fang et al. [20] and our proposed SCF-PEKS schemes require pairing operations, it would be good if an IND-KGA secure SCF-PEKS scheme can be constructed without using pairing operations. Second, it is worth to investigate on constructing an efficient SCF-PEKS scheme in a stronger security model, such as the malicious server generates her public key and secret key by herself in the security model.

Acknowledgments The authors would like to thank anonymous reviewers for their useful comments. We also thank Syh-Yuan Tan (Multimedia University) for his helpful discussions on Java implementation of the proposed scheme. L. Guo is supported by the National Science Foundation of China under Grant No. 61202365 and Technology Foundation for Selected Overseas Chinese Scholar, Department of Human Resources and Social Security of Shanxi Province. W.-C. Yau is partially supported by FRGS Grant (FRGS/1/2012/TK06/MMU/03/9).

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHIES: an encryption scheme based on the Diffie-Hellman problem. In: *CT-RSA 2001, LNCS 2020*. pp. 143–158, 2001.
2. Ateniese, G., Fu, K.V., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: *Internet Society (ISOC): NDSS 2005*. pp. 29–43, 2005.
3. Aviv, A.J., Locasto, M.E., Potter, S., Keromytis, A.D.: SSARES: Secure searchable automated remote email storage. In: *ACSAC 2007*. pp. 129–139, 2007.
4. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: *EUROCRYPT 2004, LNCS 3027*. pp. 506–522, 2004.
5. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: *ICCSA 2008, LNCS 5072*. pp. 1249–1259, 2008.
6. Baek, J., Safavi-Naini, R., Susilo, W.: On the integration of public key data encryption and public key encryption with keyword search. In: *ISW 2006, LNCS 4176*. pp. 217–232, 2006.
7. Byun, J.W., Rhee, H.S., Park, H.A., Lee, D.H.: Off -line keyword guessing attacks on recent keyword search schemes over encrypted data. In: *Proceedings of SDM 2006, LNCS 4165*. pp. 75–83, 2006.
8. Coron, J.S.: On the exact security of full domain hash. In: *Crypto 2000. LNCS 1880*. pp. 229–235, 2000.
9. Chen, Y.-C., Horng, G., Lin, Y.-J., Chen, K.-C.: Privacy preserving index for encrypted electronic medical records. *J. Med. Syst.* doi:10.1007/s10916-013-9992-x, 2013.

10. Chen, T.-S., Liu, C.-H., Cen, T.-L., Chen, C.-S., Bau, J.-G., Lin, T.-C., Secure dynamic access control scheme of PHR in cloud computing. *J. Med. Syst.* 36(6):4005–4020, 2012.
11. Chen, Y.-Y., Lu, J.-C., Jan, J.-K., A secure EHR system based on hybrid clouds. *J. Med. Syst.* 36(5):3375–3384, 2012.
12. Chen, C.-L., Yang, T.-T., Chiang, M.-L., Shih, T.-F., A privacy authentication scheme based on cloud for medical environment. *J. Med. Syst.* 38(11), 2014. doi:10.1007/s10916-014-0143-9.
13. Chen, C.-L., Yang, T.-T., Shih, T.-F., A secure medical data exchange protocol based on cloud environment. *J. Med. Syst.* 38(9), 2014. doi:10.1007/s10916-014-0112-3.
14. Chen, Y., Zhang, J., Lin, D., Zhang, Z.: Generic constructions of integrated PKE and PEKS. *Des. Codes Crypt.* http://dx.doi.org/10.1007/s10623-014-0014-x, 2014.
15. Emura, K., Miyaji, A., Omote, K.: Adaptive secure-channel free public-key encryption with keyword search implies timed release encryption. In: *ISC 2011, LNCS 7001*. pp. 102–118, 2011.
16. Emura, K., Miyaji, A., Rahman, M.S., Omote, K.: Generic constructions of secure-channel free searchable encryption with adaptive security. IACR Cryptology ePrint Archive. Available at http://eprint.iacr.org/2013/321, 2013.
17. Emura, K., and Rahman, M.S.: Constructing secure-channel free searchable encryption from anonymous IBE with partitioned ciphertext structure. In: *SECRYPT 2012*. pp. 84–93, 2012.
18. Fernández-Cardeñosa, G., de la Torre-Díez, I., López-Coronado, M., Rodrigues, J.J.P.C., Analysis of cloud-based solutions on EHRs systems in different scenarios. *J. Med. Syst.* 36(6):3777–3782, 2012.
19. Fang, L.M., Susilo, W., Ge, C.P., Wang, J.D.: A secure channel free public key encryption with keyword search scheme without random oracle. In: *CANS 2009, LNCS 5888*. pp. 248–258, 2009.
20. Fang, L.M., Susilo, W., Ge, C.P., Wang, J.D., Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.* 238:221–241, 2013.
21. Gentry, C.: Practical identity-based encryption without random oracles. In: *EUROCRYPT 2006, LNCS 4004*. pp. 445–464, 2006.
22. Kamara, S., and Lauter, K.: Cryptographic cloud storage. In: *FC 2010, LNCS 6054*. pp. 136–149, 2010.
23. Lai, J.Z., Deng, R.H., Liu, S.L.: Efficient CCA-secure PKE from Identity-based techniques. In: *CT-RSA 2010, LNCS 5985*, pp. 132–147, 2010.
24. Low, C., and Hsueh Chen, Y., Criteria for the evaluation of a cloud-based hospital information system outsourcing provider. *J. Med.l Syst.* 36(6):3543–3553, 2012.
25. Lu, C., Wu, Z., Liu, M., Chen, W., Guo, J., A patient privacy protection scheme for medical information system. *J. Med. Syst.* 37(6), 2013. doi:10.1007/s10916-013-9982-z.
26. Mat Kiah, M.L., Nabi, M.S., Zaidan, B.B., Zaidan, A.A., An enhanced security solution for electronic medical records based on AES hybrid technique with SOAP/XML and SHA-1. *J. Med. Syst.* 37(5), 2013. doi:10.1007/s10916-013-9971-2.
27. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Improved searchable public key encryption with designated tester. In: *ASIACCS 2009, ACM*. pp. 376–379, 2009.
28. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H., Trapdoor security in a searchable public key encryption scheme against keyword guessing attacks. *J. Syst. ans Softw.* 6(5):237–243, 2010.
29. Rhee, H.S., Park, J.H., Lee, D.H., Generic construction of designated tester public-key encryption with keyword search. *Inf. Sci. Express* 205(1):93–109, 2014.
30. Rhee, H.S., Susilo, W., Kim, H.J., Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electron. Express* 83:763–771, 2009.
31. Sun, J., and Fang, Y., Cross-domain data sharing in distributed electronic health record systems. *IEEE Trans. Parallel Distrib. Syst.* 21(6):754–764, 2010.
32. Susilo, W., and Win, K.T., Security and access of health research data. *J. Med. Syst.* 31(2):103–107, 2007.
33. Wang, W., Xu, P., Li, H., Yang, L.T.: Secure hybrid-indexed search for high efficiency over keyword searchable ciphertexts. *Future Gener. Comput. Syst.* doi:10.1016/j.future.2014.07.008, 2014.
34. Waters, B.: Efficient identity based encryption without random oracles. In: *EUROCRYPT 2005, LNCS 3494*. pp. 114–127. Springer-Verlag, 2005.
35. Yau, W.C., Heng, S.H., Goi, B.M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: *ATC 2008, LNCS 5060*. pp. 100–105, 2008.
36. Zhang, R., and Imai, H.: Generic combination of public key encryption with keyword search and public key encryption. In: *CANS 2007, LNCS 4856*. pp. 159–174, 2007.
37. Zhang, R., and Imai, H., Combining public key encryption with keyword search and public key encryption. *IEICE Trans.* 92-D(5):888–896, 2009.