International Journal on Cybernetics & Informatics (IJCI) Vol.13, No.2, April 2024

On the construction of Perfect Keyword Secure PEKS scheme

Indranil Ghosh Ray

Queen's University Belfast, UK

Abstract. With the growing popularity of cloud computing, searchable encryption has become centre of attraction to enhance privacy and usability of the shared data. First searchable encryption scheme under the public key setting was proposed by Bonah et al. which is known as PEKS. In the PEKS scheme, one can easily link between cipher text and the trapdoor. In Information Sciences 2017 paper, Huang et al. proposed a public key SE scheme. In this scheme, encryption of a document or keyword requires the secret key of the data sender. The data sender generates ciphertexts, and upload them onto the cloud server. The data receiver generates trapdoors depending upon the public key of the sender and its own secret key. Thus, the PEKS scheme of Huang et al. circumvents the above attack by linking the ciphertext and the trapdoor to the key of the sender. However no work is available in the literature to stop attacks against linking user key and cipher text and server key and cipher text. In this paper we address these issues. We formalize the two new security notions, namely UKI-security and SKI-security. We have shown that our scheme is secure under these newly introduced security notions.

1 Introduction

The advent of cloud computing has enabled users to outsource storage to cloud service providers. These providers offer affordable and reliable storage of client data. In addition, the cloud service provider can also perform bespoke processing on any data stored in its server by a client. Any data stored by a client on a cloud server is exposed to the risk of privacy breach. Thus, clients are more inclined to encrypt every piece of information before storing them onto a cloud server.

Encryption of stored information impedes the ability of the cloud server to perform processing of the data and drastically affects the usability of cloud services. Searchable encryption schemes were proposed to alleviate this issue. This technique enable a cloud server to search for a keyword in an encrypted document stored in its memory. For this purpose, it requires the client to issue a trapdoor corresponding to the keyword to be searched. In public key searchable encryption, the public encryption key of the data owner is used to encrypt a document, whereas, the secret key is used to generate a trapdoor. The first public key SE scheme was proposed by Boneh et al. in [4]. This scheme and almost all other public key SE schemes allow anyone having the knowledge of the public encryption key of the data owner to encrypt a document. So, given a trapdoor, the cloud server can encrypt all keywords and try to match them one by one with the trapdoor using the search algorithm. This way, the cloud server can identify the trapdoor in all circumstances.

In [14], Huang et al. proposed a public key SE scheme. In this scheme, encryption of a document or keyword requires the secret key of the data sender. This data sender generates ciphertexts, and upload them onto the cloud server. The data receiver generates trapdoors depending upon the public key of the sender and its own secret key. Thus, the PEKS scheme of Huang et al. circumvents the above attack by linking the ciphertext and the trapdoor to the key of the sender. The cloud server who does not know the secret key of the sender cannot produce ciphertexts by itself. Thus, it cannot perform a brute force search against a trapdoor.

Bibhu Dash et al: NLPAI, IOTBC -2024
pp. 67-86, 2024. IJCI – 2024 DOI:10.5121/ijci.2024.130206

Our contribution:

1. Unlike PEKS of [6], where full linkability is possible, and PSEKS scheme of [14], where partial linkability, we achieve full unlinkability in terms of user key undistinguishability (UKI) and server key undistinguishability (SKI).
2. We provide formal definition of UKI security and SKI security.
3. We provide formal security proof for UKI security and SKI security under game based modeling.
4. We implement a prototype of our scheme using java jpair library and validated against TIMIT dataset [1] which is presented in the full version of the paper. Our empirical results indicate that our scheme is feasible for practical use.

Organization The rest of the paper is organised as follows. In Section 2, we discuss previous works on phrase search.In 3 we discuss the crucial definitions that are needed. In Section 4, we describe our scheme in detail. In 5 we discuss user key indistinguishability. In Section 6, we discuss server key indistinguishability. In Section 7, we study the overhead associated with our scheme and provide a comparative study. We conclude the paper in Section 8.

2 Related Work

Searchable encryption is a newly emerging technology and has drawn attention of many leading research groups leading to considerable amount of research work in this domain [2,5,7–9,12,13,15–25] The concept of search aware encryption was first introduced by Song et al. [19]. Searchable encryption schemes susceptible to leakage of information related to *access pattern* and *search pattern*, making the scheme vulnerable to statistical attacks [12]. Curtmola et al. [11,12] proposed two schemes for keyword search in the symmetric setting. In the asymmetric setting, the entity who generates the ciphertext can delegate the search to any third party. This was introduced by Boneh et al. in [5].

In [15] Kamara et al. proposed a dynamic symmetric searchable encryption scheme which was improved by Stefanov et al. in [20]. In [8] Cash et al. proposed an encryption scheme to effectively perform dynamic search on large databases. Wang et al. proposed a fuzzy search encryption scheme that supports multiple keyword search, where the authors basically exploits a locality-sensitive hashing technique which can tolerate typos [24].

In [25] Zittrower et al. proposed the first searchable encryption for phrase search. Their construction reveals too much information to the server. In [17] it is shown that the server can learn the frequency of distinct keywords in different documents, and can know the ciphertext of likely keywords. Tang et al. proposed another construction in [23], which requires an extra lookup table, and also requires the client to store a dictionary of keywords for making search possible. Kissel et al. [16] proposed a two-phrase verifiable searchable encryption scheme. Naveed et al. in [18] proved that the leakage of the number of searched documents can be reduced by storing documents in blocks. Chase et al. [10] proposed an encryption scheme which is capable of string search. In [13], Ray et al. presented a symmetric key searchable encryption scheme that supports phrase search. In this scheme the position of the phrase in the sentence are leaked after the search. In [2], Bag et al. improved this by proposing *access pattern* secure encryption scheme where an honest-but-curious cloud server could learn nothing about even after the search.

Efficient symmetric searchable encryption scheme to support boolean search queries was proposed by Cash et al. in [9]. Li et al. [17] proposed a symmetric searchable encryption scheme that supports encrypted phrase searches known as LPSSE whose security is based

on the non-adaptive security definition of Curtmola et al [11]. Cao et al. [7] proposed a privacy-preserving multi-keyword ranked search scheme using symmetric encryption. Sun et al. [22] proposed an efficient privacy-preserving multi-keyword supporting cosine similarity measurement. Sun et al. [21] constructed an encryption scheme that supports multi-user boolean queries.

The scheme of Bonah et al. [4] is susceptible to attacking liveraging from linking between ciphertext and the trapdoor. Thus attacker can inject ciphertext of his choice and with such link, can break the trapdoor security with overwhelming probability. The PEKS scheme of Huang et al. [14] circumvents the above attack by linking the ciphertext and the trapdoor to the key of the sender only. However existing schemes are still susceptible to attacks against linking user key and cipher text and server key and cipher text. In this paper we address these issues.

3 Definition and Preliminaries

Definition 1. ***(Bilinear Pairing)*** *[6] Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear pairing, mapping from groups $\mathbb{G}_1$ and $\mathbb{G}_1$ to $\mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_1$ are cyclic groups of the same prime order p. It has the following properties.*

- *Bilinearity. For any $g, h \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, $\hat{e}(g^a, h^b) = \hat{e}(g, h)^a b$.*
- *Non-degeneracy. For any generator $g \in \mathbb{G}_1$, $\hat{e}(g, g) \in G_T$ is a generator of G_T.*
- *Computability. For any $g, h \in \mathbb{G}_1$, there is an efficient algorithm to compute $\hat{e}(g, h)$.*

Definition 2. ***(BDH)*** *Given $G, G_1, e : G \times G \rightarrow G_1$, let us define a security experiment $Exp_{\mathcal{A}}^{BDH}(\lambda)$ as follows.*

$Exp_{\mathcal{A}}^{BDH}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, c \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow e(g, g)^{abc}$
$\Omega_1 \xleftarrow{\$} G_1$
$d \xleftarrow{\$} \{0, 1\}$
$d' \leftarrow \mathcal{A}(g, g^a, g^b, g^c, \Omega_d)$
return $d = d'$

For any probabilistic polynomial time adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{BDH}(\lambda)$ is negligible, where $Adv_{\mathcal{A}}^{BDH}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{BDH}(\lambda) = 1] - \frac{1}{2} \right|$

The decision linear (DLIN) problem is defined as follows.

Definition 3. *[3]* ***(DLIN)*** *Given the group G, let us define the following security experiment $Exp_{\mathcal{A}}^{DLIN}(\lambda)$.*

$Exp_{\mathcal{A}}^{DLIN}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p$
$e \xleftarrow{\$} \mathbb{Z}_p, f \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow g^{e+f}, \Omega_1 \xleftarrow{\$} G$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{ae}, g^{f/b}, \Omega_d)$
$return\ d = d'$

The advantage of an adversary $\mathcal{A}$, against the security experiment is given by $Adv_{\mathcal{A}}^{DLIN}(\lambda)$. We define

$$Adv_{\mathcal{A}}^{DLIN}(\lambda) = \left| Pr\left[Exp_{\mathcal{A}}^{DLIN}(\lambda) = 1\right] - \frac{1}{2} \right|$$

For any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{DLIN}(\lambda) \leq negl(\lambda)$.

4 The Scheme

This scheme is an improvement over the searchable encryption scheme described in [14].

4.1 Construction

In this section, we discuss the construction of the PAEKS-II scheme. The PAEKS-II scheme is comprised of seven algorithms, namely, $Setup(\lambda)$, $SKeyGen_R(param)$, $SKeyGen_C(param)$, $SKeyGen_R(param, Pk_C)$, $Enc(param, w, Sk_S, Pk_R)$, $Trapdoor(param, w, Sk_R, Pk_R, Pk_C, Pk_S, Pk_T)$ and $Test(param, C, T_w, Pk_S, Pk_R, Sk_C)$. These algorithms are described below.

1. $Setup(\lambda)$: This algorithm takes as input the security parameter λ, and outputs true groups G, and G_1 of order p. It also returns a random generator g of G, and a hash function $H : \{0,1\}^* \rightarrow G$. Let us denote $param = (G, G_1, p, g, e, H)$.
2. $KeyGen_R(param)$: This algorithm takes as input the public parameter $param$, and it selects a random $a \in_R \mathbb{Z}_p$, and sets $(Sk_R, Pk_R) = (a, g^a)$. This algorithm returns (Sk_R, Pk_R).
3. $KeyGen_C(param)$: This algorithm takes as input the public parameter $param$, and it selects a random $c \in_R \mathbb{Z}_p$, and sets $(Sk_C, Pk_C) = (c, g^c)$. This algorithm generates a NIZK proof of knowledge of $Sk_C = \log_g Pk_C$. This algorithm returns (Sk_C, Pk_C, Π_C).
4. $KeyGen_S(param, Pk_C)$: This algorithm takes as input the public parameter $param$, and the public key of the cloud Pk_C. It selects a random $b \in \mathbb{Z}_p$, and sets $(Sk_S, Pk_S, Pk_T) = (b, g^b, Pk_C^b)$. This algorithm returns (Sk_S, Pk_S, Pk_T).
5. $Enc(param, w, Sk_S, Pk_R)$: This algorithm takes as input the public parameter $param$, a keyword w, the secret key Sk_S of the data sender, and the public key Pk_R of the data receiver. The algorithm selects a random $r \in_R \mathbb{Z}_p$, and computes $C_1 = H(w)^{Sk_S} \cdot g^r, C_2 = Pk_R^r$. This algorithm returns the ciphertext $C = (C_1, C_2)$.
6. $Trapdoor(param, w, Sk_R, Pk_R, Pk_C, Pk_S, Pk_T)$: This algorithm takes as input the public parameter $param$, a keyword w the secret key Sk_R of the receiver, the public keys Pk_C, and Pk_S of the cloud and the data sender, and the public trapdoor generation key Pk_T. It selects random $R \in_R G$, and computes $T_w = \big(e(Pk_S, H(w)^{Sk_R} R), e(Pk_T, R)\big)$. The algorithm returns the trapdoor T_w.

7. $Test(param, C, T_w, Pk_S, Pk_R, Sk_C)$: This algorithm takes as input the public parameter $param$, the ciphertext $C = (C_1, C_2)$, the trapdoor $T_w = (A, B)$, the public keys Pk_S, Pk_R of the sender and the receiver, and the secret key Sk_C of the cloud. This algorithm returns 1 if $(A/B^{1/Sk_C}) \cdot e(C_2, g) = e(C_1, Pk_R)$, or returns 0 otherwise.

4.2 Correctness

We show that given a ciphertext, and a correctly generated trapdoor, the $Test(\cdot)$ algorithm will return either 1 if the ciphertext and the trapdoor correspond to the same keyword, or 0 otherwise. The ciphertext $C = (C_1, C_2)$, where $C_1 = H(w)^{Sk_S} \cdot g^r$, and $C_2 = Pk_R^r$.
Again, the trapdoor is $T_w = \big(e(Pk_S, H(w)^{Sk_R}R), e(Pk_T, R)\big)$.
Now, $\big(e(Pk_S, H(w)^{Sk_R}R)/e(Pk_T, R)^{Sk_C}\big) \cdot e(C_2, g) = \big(e(Pk_S, H(w)^{Sk_R}R)/e(Pk_S, R)\big) \cdot e(Pk_R^r, g) = e(Pk_S, H(w)^{Sk_R}) \cdot e(Pk_R, g^r) = e(Pk_R, H(w)^{Sk_S}) \cdot e(Pk_R, g^r) = e(Pk_R, H(w)^{Sk_S} g^r) = e(Pk_R, C_1)$.

Now, let us assume that the ciphertext correspond to the word w, and the trapdoor correspond to the word w'. That is, the ciphertext is $C = (C_1, C_2) = (H(w)^{Sk_S} g^r, Pk_R^r)$, and the trapdoor is $T_w = \big(e(Pk_S, H(w')^{Sk_R}R), e(Pk_T, R)\big)$. As such, $(e(Pk_S, H(w')^{Sk_R}R)/e(Pk_T, R)^{Sk_C}) \cdot e(Pk_R^r, g) = e(Pk_S, H(w')^{Sk_R}) \cdot e(Pk_R^r, g) = e(Pk_R, H(w')^{Sk_S}) \cdot e(Pk_R, g^r) = e(Pk_R, H(w')^{Sk_S} \cdot g^r) \neq e(C_2, g)$. Thus, our scheme is correct.

4.3 Keyword Privacy

The first security notion which we discuss is called 'keyword privacy'. This security notion enables us to understand whether a PEKS scheme can resist leakage of information about the keyword through ciphertexts and trapdoors. In other words, a PEKS scheme will be secure according to this notion if the adversary cannot extract any useful keyword-related information from a pair of trapdoor and ciphertext corresponding to the same keyword. We formally define this notion below.

Let us consider the following security experiment $Exp_{\mathcal{A}}^{KWP}(\lambda)$. We denote by $Adv_{\mathcal{A}}^{KWP}(\lambda)$, the advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ against the experiment $Exp_{\mathcal{A}}^{KWP}(\lambda)$. We define $Adv_{\mathcal{A}}^{KWP}(\lambda) = \big|Pr[Exp_{\mathcal{A}}^{KWP}(\lambda) = 1] - \frac{1}{2}\big|$. A PEKS scheme is secure in accordance with the keyword privacy notion if for any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{KWP}(\lambda) \leq negl(\lambda)$.

$Exp_{\mathcal{A}}^{KWP}(\lambda)$
$(G, G_1, p, g, e, H) \leftarrow Setup(\lambda)$
$(Sk_R, Pk_R) \leftarrow KeyGen_R(G, G_1, p, g, e, H)$
$(st_1, Pk_C, \Pi_C) \leftarrow \mathcal{A}_0^{KeyGen_C}(G, G_1, p, g, e, H, Pk_R)$
$(Sk_S, Pk_S, Pk_T) \leftarrow KeyGen_S(G, G_1, p, g, e, H, Pk_C)$
$(w_0^*, w_1^*, \tilde{Q}, st_2) \leftarrow \mathcal{A}_1^{Enc(\cdot), Trapdoor(\cdot)}(st_1, Pk_S, Pk_T)$
If $Enc(\cdot, w_i^*, \cdot)$ or $Trapdoor(\cdot, w_i^*, \cdot) : i \in \{0, 1\}$ is queried by $\mathcal{A}_0$
return 0
$d \xleftarrow{\$} \{0, 1\}$
$X_i = Enc(\cdot, w_d^*, \cdot), Y_i = Trapdoor(\cdot, w_d^*, \cdot) : i \in [1, \tilde{Q}]$
$X = \{X_i : i \in [1, \tilde{Q}]\}, Y = \{Y_i : i \in [1, \tilde{Q}]\}$
$d' \leftarrow \mathcal{A}_2^{Enc(\cdot), Trapdoor(\cdot)}(st_2, X, Y)$
If $Enc(\cdot, w_i^*, \cdot)$ or $Trapdoor(\cdot, w_i^*, \cdot) : i \in \{1, 2\}$ is queried by $\mathcal{A}_1$
return 0
else return $d = d'$

In the above security experiment, the challenger first uses the *Setup* function to generate the public parameters. Then it generates the keys of the receiver, server, and the sender. There is a two stage adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$. Now, it invokes $\mathcal{A}_0$ with the public parameters and the public keys. $\mathcal{A}_0$ is given access to the Encryption and Trapdoor oracles. $\mathcal{A}_0$ outputs two keywords, such that $\mathcal{A}_0$ has not made an encryption or trapdoor query for any of them. $\mathcal{A}_0$ also outputs an integer $Q \in poly(\lambda)$. The challenger randomly picks one of the keywords, and computes a ciphertext, and trapdoor for that particular keyword. Then the challenger invokes $\mathcal{A}_1$ with the ciphertext and trapdoor. The adversary has to guess which keyword the ciphertext and the trapdoor correspond to. If the adversary makes a correct guess, it wins the experiment.

Here we provide a varint of DLIN defined in definition 3

Assumption 1 *Given the group G, let us define the following security experiment $Exp_{\mathcal{A}}^{DLIN1}(\lambda)$.*

$Exp_{\mathcal{A}}^{DLIN1}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p$
$e \xleftarrow{\$} \mathbb{Z}_p, f \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow g^{f/b}, \Omega_1 \xleftarrow{\$} G$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{ae}, g^{e+f}, \Omega_d)$
return $d = d'$

The advantage of an adversary $\mathcal{A}$, against the security experiment is given by $Adv_{\mathcal{A}}^{DLIN1}(\lambda)$. We define

$$Adv_{\mathcal{A}}^{DLIN1}(\lambda) = \left| Pr\left[Exp_{\mathcal{A}}^{DLIN1}(\lambda) = 1\right] - \frac{1}{2} \right|$$

For any PPT adversary $\mathcal{A}$, $Adv_{\mathcal{A}}^{DLIN1}(\lambda) \leq negl(\lambda)$.

Lemma 1. *Definition 3 implies assumption 1.*

Proof. We show that if there exists an adversary $\mathcal{A}$, against the security experiment $Exp_{\mathcal{A}}^{DLIN1}(\lambda)$, it could be used to construct another adversary $\mathcal{B}$, against the security experiment $Exp_{\mathcal{B}}^{DLIN}(\lambda)$. $\mathcal{B}$ works as follows. It receives as input $g, g^a, g^{ae}, g^b, g^{b/f}$, and a challenge $\Omega_d \in \{g^{e+f}, R\}$, where R is uniformly random in G. $\mathcal{B}$ invokes $\mathcal{A}$ with these inputs: $g, g^a, g^{ae}, g^b, \Omega_d, g^{f/b}$. Here, the challenge is $\omega_d = g^{e+f}$. Observe that, if $\Omega_d = g^{e+f}$, then $\omega_d = g^{f/b}$, alternatively, if Ω_d is random, Ω_d is also random. Hence, if $\mathcal{A}$ can identify Ω_d, $\mathcal{B}$ will be able to identify Ω_d correctly. Hence, the lemma holds. □

Assumption 2 *Given $G, G_1, e : G \times G \to G_1$, for all probabilistic polynomial time adversary $\mathcal{A}$, the following probability is negligibly higher than $\frac{1}{2}$.*

$$Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p, f \xleftarrow{\$} \mathbb{Z}_p \\ \Omega_0 \leftarrow g^{f/b}, \Omega_1 \xleftarrow{\$} G \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{ae}, e(g,g)^{af}, g^{e+f}, \Omega_d) \end{array}\right]$$

Lemma 2. *Assumption 1 implies assumption 2.*

Proof. We show that if there exists an adversary $\mathcal{A}$, against the assumption 2, it could be used in the construction of another adversary $\mathcal{B}$, against the assumption 1. $\mathcal{B}$ works as follows: it receives as input $g, g^a, g^b, g^{ae}, g^{e+f}$, and the challenge $\Omega_d \in_R \{g^{f/b}, R\}$, where R is uniformly random in G. $\mathcal{B}$ computes $e(g,g)^a f$ as $e(g^a, g^{e+f})/e(g, g^{ae})$. Now, $\mathcal{B}$ invokes $\mathcal{A}(g, g^a, g^b, g^{ae}, e(g,g)^{af}, g^{e+f}, \Omega_d)$, and it returns what $\mathcal{A}$ returns. It is easy to see that the lemma holds. □

Assumption 3 *Given $G, G_1, e : G \times G \to G_1$, for all probabilistic polynomial time adversary $\mathcal{A}$, the following probability is negligibly higher than $\frac{1}{2}$.*

$$Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ \Omega_0 \leftarrow g^k, \Omega_1 \xleftarrow{\$} G \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{ae}, e(g,g)^{abk}, g^{e+bk}, \Omega_d) \end{array}\right]$$

Lemma 3. *Assumption 2 implies assumption 3.*

Proof. One can reduce the former assumption to the latter one by setting $k = f/b$. Hence, the lemma holds. □

Assumption 4 *Given $G, G_1, e : G \times G \to G_1$, for all probabilistic polynomial time adversary $\mathcal{A}$, the following probability is negligibly higher than $\frac{1}{2}$.*

$$Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k_0 \xleftarrow{\$} \mathbb{Z}_p, k_1 \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{ae}, e(g,g)^{abk_d}, g^{e+bk_d}, g^{k_0}) \end{array}\right]$$

Lemma 4. *Assumption 3 implies assumption 4.*

Proof. We show that if there exists an adversary $\mathcal{A}$, against assumption 4, it could be used to construct another adversary $\mathcal{B}$, against assumption 3. $\mathcal{B}$ receives as input $g, g^a, g^b, g^{ae}, e(g,g)^{abk}, g^{e+bk}$, and $\Omega_d \in \{g^k, R\}$, where R is uniformly random in G. $\mathcal{B}$ implicitly sets $k_0 = \log_g \Omega_d$, and invokes
$\mathcal{A}(g, g^a, g^b, g^{ae}, e(g,g)^{abk}, g^{e+bk}, g^{k_0})$. It is easy to see that if $\mathcal{A}$ can distinguish between the two possible cases, so can $\mathcal{B}$. Hence, the lemma holds. □

Assumption 5 *Given $G, G_1, e : G \times G \to G_1$, for all probabilistic polynomial time adversary $\mathcal{A}$, the following probability is negligibly higher than $\frac{1}{2}$.*

$$Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k_0 \xleftarrow{\$} \mathbb{Z}_p, k_1 \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{k_0}, g^{k_1}, g^{ae}, e(g,g)^{abk_d}, g^{e+bk_d}) \end{array}\right]$$

Lemma 5. *Assumption 4 implies assumption 5.*

Proof. Let us assume that

$$X = Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k_0 \xleftarrow{\$} \mathbb{Z}_p, k_1 \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{k_0}, g^{k_1}, g^{ae}, e(g,g)^{abk_d}, g^{e+bk_d}) \end{array}\right]$$

$$A = Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k_0 \xleftarrow{\$} \mathbb{Z}_p, k_1 \xleftarrow{\$} \mathbb{Z}_p, k_2 \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ d \xleftarrow{\$} \{0,2\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{k_0}, g^{k_1}, g^{ae}, e(g,g)^{abk_d}, g^{e+bk_d}) \end{array}\right]$$

and

$$B = Pr\left[\begin{array}{c} g \xleftarrow{\$} G \\ a \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \mathbb{Z}_p, k_0 \xleftarrow{\$} \mathbb{Z}_p, k_1 \xleftarrow{\$} \mathbb{Z}_p, k_2 \xleftarrow{\$} \mathbb{Z}_p, e \xleftarrow{\$} \mathbb{Z}_p \\ d \xleftarrow{\$} \{1,2\} \\ d' \leftarrow \mathcal{A}(g, g^a, g^b, g^{k_0}, g^{k_1}, g^{ae}, e(g,g)^{abk_d}, g^{e+bk_d}) \end{array}\right]$$

From triangular inequality, we can say that $A + B \geq X$. Again, from assumption 3, we can say that $A = B$. Hence, the result holds. □

Theorem 1. *Our PEKS scheme is secure according to the Keyword Privacy notion.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, against our PEKS scheme, then using it, one can construct another adversary $\mathcal{B}$ against the assumption 5. $\mathcal{B}$ works as follows:

$\mathcal{B}$ receives as inputs $g, g^a, g^b, g^{k_0}, g^{k_1}, g^{ae}$, and the challenge $\Omega_d = (\Omega_{d1}, \Omega_{d2}) = (e(g,g)^{abk_d}, g^{e+bk_d})$. Here, $d \in_R \{0,1\}$. $\mathcal{B}$ implicitly sets $Pk_S = g^b$, and $Pk_R = g^a$. First $\mathcal{B}$ invokes $\mathcal{A}_0$ with all the required inputs, and receives Pk_C, and Π_C. $\mathcal{B}$ receives Sk_C from the simulation of $\mathcal{A}_0$ or by extracting the NIZK proof Π_C. $\mathcal{B}$ invokes $\mathcal{A}_1$ with the necessary inputs. Let, $\mathbf{K} = \{g^{k_0}, g^{k_1}\}$. $\mathcal{B}$ answers all queries of $\mathcal{A}_1$ to the oracles $Enc(\cdot)$, and $Trapdoor(\cdot)$. Apart from these, $\mathcal{B}$ also needs to answer the queries to the hash oracle $H(\cdot)$. The hash queries can be made either directly by $\mathcal{A}_1$, or through an encryption/trapdoor query by $\mathcal{A}_1$. Whenever, $\mathcal{A}_1$ makes a query to the $Enc(\cdot)$ oracle with the input w, $\mathcal{B}$ selects random $x \in_R \mathbb{Z}_p$, and sets $H(w) = g^x$. $\mathcal{B}$ adds $\langle w, x, g^x \rangle$ to list $\mathbf{H}$. $\mathcal{B}$ also selects random $r \in_R \mathbb{Z}_p$, and computes $C_1 = Pk_S^x g^r$, and $C_2 = Pk_R^r$. It then returns $Enc(w) = (C_1, C_2)$. If $\mathcal{A}_1$ makes a trapdoor query for a keyword w, then $\mathcal{B}$ selects a random $y \in_R \mathbb{Z}_p$, and computes $H(w) = g^y$. $\mathcal{B}$ adds $\langle w, y, g^y \rangle$ to list $\mathbf{H}$. $\mathcal{B}$ also selects random $\tilde{r} \in_R \mathbb{Z}_p$, and computes $T_1 = e(Pk_S, Pk_R^y g^{\tilde{r}})$, and $T_2 = e(Pk_T, g^{\tilde{r}})$. $\mathcal{B}$ returns $T(w) = (T_1, T_2)$. Whenever $\mathcal{A}$ makes a direct query to the $H(\cdot)$ oracle for a keyword w, $\mathcal{B}$ selects a random element h from G. If $\mathbf{K} = \emptyset$, $\mathcal{B}$ assigns $H(w) = h$. Else, it selects $\delta \in_R \mathbf{K}$, and assigns

$$H(w) = \begin{cases} h \text{ with probability } 1/Q \\ g^{k_t} \text{ with probability } 1 - 1/Q \end{cases}$$

$\mathcal{B}$ updates $\mathbf{K}$ as $\mathbf{K} = \mathbf{K} \setminus \{H(w)\}$. $\mathcal{B}$ returns $H(w)$.

Now, $\mathcal{A}_1$ should output the two challenge keywords: $\tilde{w}_0$, and $\tilde{w}_1$. As required, $\mathcal{A}_1$ must not have requested for either an encryption query or a trapdoor query for either of them. $\mathcal{A}_1$ also returns $Pk_C \in G$. However, $\mathcal{A}_1$ could have made hash query against one or both the challenge-keywords. If none of them were involved in a previous hash query, then $\mathcal{B}$

assigns $H(\tilde{w}_0) = g^{k_0}$, and $H(\tilde{w}_1) = g^{k_1}$. Else, if there exists $l \in \{0,1\}$, such that $H(\tilde{w}_l)$ was not queried, then assign $H(\tilde{w}_l) = g^{k_l}$. Now, if $\{H(\tilde{w}_0), H(\tilde{w}_1)\} \neq \{H(g^{k_0}), H(g^{k_1})\}$, then $\mathcal{B}$ aborts and returns a random bit. Else, $\mathcal{B}$ generates $\tilde{Q}$ ciphertexts as $C_1, C_2, \ldots, C_{\tilde{Q}}$. Here, $C_i = (C_{1i}, C_{2i})$, where $C_{1i} = \Omega_{d2} * g^{\alpha_i}, C_{2i} = g^{ae} * (g^a)^{\alpha_i} : i \in [1, \tilde{Q}]$, where $\alpha_i \in_R \mathbb{Z}_p$. $\mathcal{B}$ also generates $\tilde{Q}$ trapdoors from $e(g,g)^{abk_d}$. It selects random $\beta_i \in_R \mathbb{Z}_p$ for $i \in [1, \tilde{Q}]$, and computes $T_{1i} = e(g,g)^{abk_d} \cdot e(Pk_S, g)^{\beta_i}$, and $T_{2i} = e(Pk_C, Pk_S)^{\beta_i}$, for all $i \in [1, \tilde{Q}]$. Let T_{wi} denote (T_{1i}, T_{2i}), for all $i \in [1, \tilde{Q}]$. Now, $\mathcal{B}$ invokes $\mathcal{A}_2$ with all the necessary inputs, including the $\tilde{Q}$ ciphertexts and trapdoors. $\mathcal{B}$ answers the queries of $\mathcal{A}_2$ to the encryption and trapdoor oracles as it did for similar queries from $\mathcal{A}_1$. $\mathcal{B}$ responds to all the hash queries of $\mathcal{A}_2$ by returning randomly sampled elements from G. If $\mathcal{A}_2$ can identify the value of d, so can $\mathcal{B}$.

Let us now calculate the success probability of $\mathcal{B}$. $\mathcal{B}$ returns a random bit when it needs to abort. Thus, if it aborts, its success probability is $\frac{1}{2}$. If it does not abort, its success probability is same as that of $\mathcal{A}$. $\mathcal{B}$ aborts only when $\{H(\tilde{w}_0), H(\tilde{w}_1)\} \neq \{H(g^{k_0}), H(g^{k_1})\}$. This happens when $\mathcal{A}_1$, makes a direct query $H(\tilde{w}_j)$, but does not receive g^{k_0} or g^{k_1}, for some $j \in \{0,1\}$. Hence, we can write $Pr[Exp_{\mathcal{B}}^{ASS6}(\lambda) = 1] \geq Pr[\mathcal{B} \text{ aborts }] * \frac{1}{2} + Pr[\mathcal{B} \text{ does not abort }] * Pr[Exp_{\mathcal{A}}^{KWP}(\lambda)] = \frac{1}{2} + Pr[\mathcal{B} \text{ does not abort }] * Adv_{\mathcal{A}}^{KWP}(\lambda)$. Now, the probability $Pr[\mathcal{B} \text{ does not abort }]$ is at least $\mathbf{F} = \min_{i=0}^{Q-2}(1 - \frac{1}{Q})^i \frac{1}{Q^2}$. As such, $\mathbf{F} \geq \frac{1}{Q^2}(1 - \frac{1}{Q})^{Q-2} \geq \frac{1}{4(Q-1)^2}$. Thus, $Adv_{\mathcal{A}}^{KWP}(\lambda) \leq 4(Q-1)^2 * Adv_{\mathcal{B}}^{ASS6}(\lambda)$. Hence, the result holds. □

4.4 Privacy of Trapdoor

We now discuss a different privacy property of our scheme which is called privacy of trapdoor. We show that our scheme does not allow an adversary to learn any information from the trapdoor. For this purpose, let us consider the following security experiment.

$Exp_{\mathcal{A}}^{SPP}(\lambda)$
$(G, G_1, p, g, e, H) \leftarrow Setup(\lambda)$
$(Sk_R, Pk_R) \leftarrow KeyGen_R(G, G_1, p, g, e, H)$
$(Pk_C, \Pi_C) \leftarrow KeyGen_C(G, G_1, p, g, e, H)$
$(st_1, Pk_S, Pk_T) \leftarrow \mathcal{A}_0^{KeyGen_S}(G, G_1, p, g, e, H, Pk_R, Pk_C, \Pi_C)$
if $e(Pk_T, g) \neq e(Pk_C, Pk_S)$
return 0
$(w_0^*, w_1^*, st_2) \leftarrow \mathcal{A}_1^{Trapdoor(\cdot)}(st_1)$
$d \xleftarrow{\$} \{0,1\}$
$T_w = Trapdoor(\cdot, w_d^*, \cdot)$
$d' \leftarrow \mathcal{A}_2^{Trapdoor(\cdot)}(st_2, T_w)$
return $d = d'$

In this above security experiment, the challenger generates the public and secret keys of the receiver and the cloud server. The adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ emulates the sender. First, $\mathcal{A}_0$ generates the public keys for the sender. Then $\mathcal{A}_1$ is invoked by the challenger. It produces two challenge keywords: w_0^*, and w_1^*. $\mathcal{A}_1$ can query the trapdoor oracle for any keyword. The challenger flips a coin, and depending upon the output, chooses either w_0^*, or w_1^*. Then, the challenger generates a trapdoor T_w for the chosen keyword. Now, the challenger invokes $\mathcal{A}_2$ with the trapdoor T_w. $\mathcal{A}_2$ has to guess whether T_w corresponds to w_0^*, or w_1^*. The advantage of the adversary $\mathcal{A}$, against the security experiment $Exp_{\mathcal{A}}^{SPP}(\lambda)$ is denoted as $Adv_{\mathcal{A}}^{SPP}(\lambda)$. We define it as,

$$Adv_{\mathcal{A}}^{SPP}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{SPP}(\lambda) = 1] - \frac{1}{2} \right|$$

We say that the PEKS scheme provides SPP security, if for all probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, $Adv_{\mathcal{A}}^{SPP}(\lambda) \leq negl(\lambda)$. Note that in this adversarial model, we wanted the adversary to have the same knowledge, and capability as that of the data sender. This is because, if the scheme offers SPP security when the data sender is acting as the adversary, the scheme will obviously be secure as per the SPP security notion against any other adversary than the cloud server.

Assumption 6 *Given $G, G_1, e : G \times G \rightarrow G_1$, for all PPT adversary $\mathcal{A}$, the following probability is negligible higher than $\frac{1}{2}$.*

$$Pr\begin{bmatrix} g \xleftarrow{\$} G, \Delta \xleftarrow{\$} G_1 \\ a \xleftarrow{\$} \mathbb{Z}_p \\ \Omega_0 \leftarrow \Delta^a, \Omega_1 \xleftarrow{\$} G_1 \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, \Delta, \Omega_d) \\ return d{=}d' \end{bmatrix}$$

Lemma 6. *Assumption 2 implies assumption 6.*

Proof. We show that if there exists an adversary $\mathcal{A}$, against the assumption 6, it could be used in the construction of another adversary $\mathcal{B}$ against assumption 2. $\mathcal{B}$ receives as input g^a, g^b, g^c, and a challenge $\Omega_d \in_R \{e(g,g)^{abc}, R\}$, where $R \in_R \mathbb{Z}_p$. $\mathcal{B}$ computes $\Delta = e(g^a, g^b)$. Now, $\mathcal{B}$ invokes $\mathcal{A}(g, g^a, \Delta, \Omega_d)$. $\mathcal{A}$ returns a bit d'. $\mathcal{B}$ returns the same bit. It is easy to see that the lemma holds. □

Assumption 7 *Given $G, G_1, e : G \times G \rightarrow G_1$, for all PPT adversary $\mathcal{A}$, the following probability is negligible higher than $\frac{1}{2}$.*

$$Pr\begin{bmatrix} g \xleftarrow{\$} G, \Delta \xleftarrow{\$} G_1 \\ a \xleftarrow{\$} \mathbb{Z}_p \\ \Omega_0 \leftarrow \Delta, \Omega_1 \xleftarrow{\$} G_1 \\ d \xleftarrow{\$} \{0,1\} \\ d' \leftarrow \mathcal{A}(g, g^a, \Delta^a, \Omega_d) \\ return d{=}d' \end{bmatrix}$$

Lemma 7. *Assumption 7 implies assumption 6.*

Proof. We show that if there exists an adversary $\mathcal{A}$, against the assumption 7, one can use it to construct another adversary $\mathcal{B}$, against the assumption 6. $\mathcal{B}$ receives as inputs the following items: g, g^a, Δ, and Ω_d. It invokes $\mathcal{A}(g, g^a, \Omega_d, \Delta)$.Note that, if $\Omega_d \neq \Delta^a$, then Δ should appear as a random element of G_1 to $\mathcal{A}$. Hence, if $\mathcal{A}$ can distinguish between the two cases, $\mathcal{B}$ can identify Ω_d. Hence, the lemma holds. □

Assumption 8 *Given $G, G_1, e : G \times G \rightarrow G_1$, let us consider the following security experiment $Exp_{\mathcal{A}}^{TRD}(\lambda)$.*

$Exp_{\mathcal{A}}^{TRD}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p$
$g^k \leftarrow \mathcal{A}_0(G, G_1, e, g, g^a)$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow e(g^k, g^r), \Omega_1 \xleftarrow{\$} G_1$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}_1(e(g^k, g^r)^a, \Omega_d)$
$return\ d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{TRD}(\lambda)$ is given by

$$Adv_{\mathcal{A}}^{TRD}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD}(\lambda) = 1] - \frac{1}{2} \right|$$

For all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $Adv_{\mathcal{A}}^{TRD}(\lambda) \leq negl(\lambda)$.

Lemma 8. *Assumption 7 implies assumption 8.*

Proof. Let us assume that there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{TRD}(\lambda)$. We show how another adversary $\mathcal{B}$ can be constructed using $\mathcal{A}$. $\mathcal{B}$ receives as inputs, g, g^a, Δ^a, and Ω_d. $\mathcal{B}$ invokes $\mathcal{A}$, and receives g^k. $\mathcal{B}$ implicitly assigns $\Delta = e(g^k, g^r)$. If $\mathcal{A}_1$ can identify whether $\Omega_d = \Delta$ or Ω_d is random, so can $\mathcal{B}$. Hence, the lemma holds. □

Assumption 9 *Given $G, G_1, e : G \times G \rightarrow G_1$, let us consider the following security experiment $Exp_{\mathcal{A}}^{TRD1}(\lambda)$.*

$Exp_{\mathcal{A}}^{TRD1}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p$
$b_0 \xleftarrow{\$} \mathbb{Z}_p, b_1 \xleftarrow{\$} \mathbb{Z}_p$
$c \xleftarrow{\$} \mathbb{Z}_p$
$g^k \leftarrow \mathcal{A}_0(G, G_1, e, g, g^a, g^{b_0}, g^{b_1}, g^c)$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow e(g^k, g^{c \cdot b_0 + r}), \Omega_1 \xleftarrow{\$} e(g^k, g^{c \cdot b_1 + r})$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}_1(g, g^a, e(g^k, g^r)^a, \Omega_d)$
$return\ d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{TRD1}(\lambda)$ is given by

$$Adv_{\mathcal{A}}^{TRD1}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1] - \frac{1}{2} \right|$$

For all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $Adv_{\mathcal{A}}^{TRD1}(\lambda) \leq negl(\lambda)$.

Lemma 9. *Assumption 8 implies assumption 9.*

Proof. Let us consider another security experiment $Exp_{\mathcal{A}}^{TRD2}(\lambda)$ as follows.

$Exp_{\mathcal{A}}^{TRD2}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p$
$b_0 \xleftarrow{\$} \mathbb{Z}_p, b_1 \xleftarrow{\$} \mathbb{Z}_p$
$c \xleftarrow{\$} \mathbb{Z}_p$
$g^k \leftarrow \mathcal{A}_0(G, G_1, e, g, g^a, g^{b_0}, g^{b_1}, g^c)$
$r \xleftarrow{\$} \mathbb{Z}_p$
$R \xleftarrow{\$} G_1$
$\Omega_0 \leftarrow e(g^k, g^{c \cdot b_0}) * R, \Omega_1 \xleftarrow{\$} e(g^k, g^{c \cdot b_1}) * R$
$d \xleftarrow{\$} \{0,1\}$
$d' \leftarrow \mathcal{A}_1(g, g^a, e(g^k, g^r)^a, \Omega_d)$
return $d = d'$

Let $Adv_{\mathcal{A}}^{TRD2}(\lambda)$ be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against $Exp_{\mathcal{A}}^{TRD2}(\lambda)$. Let us define it as

$$Adv_{\mathcal{A}}^{TRD2}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1] - \frac{1}{2} \right|$$

Now, $Adv_{\mathcal{A}}^{TRD1}(\lambda) + Adv_{\mathcal{A}}^{TRD2}(\lambda) = Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1] + Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1] - 1 = (Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 0] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 1]) + (Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 0] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 1]) - 1 = \frac{1}{2}(Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 0] + Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 0]) + \frac{1}{2}(Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 1] + Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 1]) - 1$. It is easy to see that $Pr[Exp_{\mathcal{A}}^{TRD}(\lambda) = 1 | d = 0] = Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 0] = Pr[Exp_{\mathcal{A}}^{TRD1}(\lambda) = 1 | d = 1]$, and $Pr[Exp_{\mathcal{A}}^{TRD}(\lambda) = 1 | d = 1] = Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 0] = Pr[Exp_{\mathcal{A}}^{TRD2}(\lambda) = 1 | d = 1]$. Hence, $Adv_{\mathcal{A}}^{TRD1}(\lambda) + Adv_{\mathcal{A}}^{TRD2}(\lambda) \leq Pr[Exp_{\mathcal{A}}^{TRD}(\lambda) = 1] + Pr[Exp_{\mathcal{A}}^{TRD}(\lambda) = 1] - 1 = 2 * Adv_{\mathcal{A}}^{TRD}(\lambda)$. Thus, $Adv_{\mathcal{A}}^{TRD1}(\lambda) \leq 2 * Adv_{\mathcal{A}}^{TRD}(\lambda)$. Hence, the lemma holds. □

Lemma 10. *Our PEKS scheme is SPP secure.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, against the security experiment $Exp_{\mathcal{A}}^{SPP}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$, against the security experiment $Exp_{\mathcal{A}}^{TRD1}(\lambda)$. $\mathcal{B}$ functions as follows: first $\mathcal{B}_0$ receives as input $G, G_1, e, g^a, g^{b_0}, g^{b_1}$, and g^c. $\mathcal{B}_0$ implicitly sets $(Sk_R, Pk_R = c, g^c)$, and $(Sk_C, Pk_C) = (a, g^a)$. $\mathcal{B}$ generates Π_C, the simulated NIZK proof of knowledge of Sk_C, given Pk_C. $\mathcal{B}_0$ invokes $\mathcal{A}_0$ with the necessary inputs. It generates the keys (Pk_S, Pk_T). $\mathcal{B}$ checks if $e(Pk_T, g) = e(Pk_C, Pk_S)$. $\mathcal{B}_0$ learns Sk_S from the simulation of $\mathcal{A}_0$. Then $\mathcal{B}_0$ invokes $\mathcal{A}_1$. $\mathcal{B}_0$ answers all the queries of $\mathcal{A}_1$ to the oracles $Trapdoor(\cdot)$, and $H(\cdot)$. The hash oracle $H(\cdot)$ can be queried by $\mathcal{A}$ in two ways; 1) either directly or 2) indirectly through a trapdoor query. Let X be $\{g^{b_0}, g^{b_1}\}$. Whenever a $H(w)$ query is made (directly/indirectly) for some trapdoor w, $\mathcal{B}_0$, samples random $\alpha \in_R \mathbb{Z}_p$, and computes $\mathbf{A} = g^\alpha$. If $X = \emptyset$, $\mathcal{B}_0$ returns $\mathbf{Y} = g^\alpha$. Else, $\mathcal{B}_0$ selects $x \in_R X$, and returns $\mathbf{Y}$, where

$$\mathbf{Y} = \begin{cases} x & \text{with probability } 1/Q \\ \mathbf{A} & \text{with probability } 1 - 1/Q \end{cases}$$

Here, Q is the maximum number of queries made to the oracle $H(\cdot)$, including the direct and indirect ones. $\mathcal{B}_0$ adds $\langle w, \mathbf{Y}\rangle$ to L_H. If x is returned, X is updated as $X = X \setminus \{x\}$. When a trapdoor query is made by $\mathcal{A}_1$ for some keyword w, $\mathcal{B}_0$ selects random $l \in \mathbb{Z}_p$, and computes $T_{w1} = e(Pk_R, H(w))^{Sk_S} * e(Pk_S, g^l)$, and $T_{w2} = e(Pk_T, g^l)$. It then returns $T_w = (T_{w1}, T_{w2})$. Now, $\mathcal{A}_1$ returns two keywords: w_0^*, and w_1^*. $\mathcal{B}_0$ returns Pk_S. If for some $\beta \in \{0, 1\}$, $H(w_\beta^*)$ was queried but neither g^{b_0} or g^{b_1} was returned, $\mathcal{B}_0$ sends instruction to $\mathcal{B}_1$ to return a random bit. This instruction is passed through the auxiliary variable st_2. Otherwise $\mathcal{B}_0$ makes sure that $\{H(w_0^*), H(w_1^*)\} = \{g^{b_0}, g^{b_1}\}$ holds.

When $\mathcal{B}_1$ is invoked with the inputs $e(g^k, g^r)^a$, and Ω_d, it invokes $\mathcal{A}_2(st_2, \Omega_d, e(g^k, g^r)^a)$. $\mathcal{A}_2$'s queries to the trapdoor and the hash oracles are responded in the same as was being done for $\mathcal{A}_1$. It is easy to see that if $\mathcal{A}_2$ can identify whether the trapdoor corresponds to w_0^*, or w_1^*, $\mathcal{B}_1$, can identify d.

We now calculate the success probability of $\mathcal{B}$. If $\mathcal{B}_1$ aborts, then it returns a random bit. Thus, if $\mathcal{B}$ aborts, its success probability would be $\frac{1}{2}$. If it does not abort, then its success probability will be same as that of $\mathcal{A}$, which is $Pr[Exp_{\mathcal{A}}^{SPP}(\lambda) = 1]$. $\mathcal{B}_1$ has to abort if $\{g^{w_0^*}, g^{w_1^*}\} \neq \{g^{b_0}, g^{b_1}\}$. Now, we know that $\mathcal{B}_1$ aborts if $\mathcal{A}_1$ queries $H(w_0^*)$ or $H(w_1^*)$, but g^{b_0} or g^{b_1} is not returned. The probability of this event not happening is at least $\frac{1}{Q^2}(1 - \frac{1}{Q})^{Q-2}$. Thus, the probability that $\mathcal{B}$ will succeed is $Pr[Exp_{\mathcal{B}}^{TRD1}(\lambda) = 1] \geq \frac{1}{2} * Pr[\mathcal{B} \text{ aborts }] + Pr[\mathcal{B} \text{ does not abort }] * Pr[Exp_{\mathcal{A}}^{SPP}(\lambda)] = \frac{1}{2} * Pr[\mathcal{B} \text{ aborts }] + Pr[\mathcal{B} \text{ does not abort }] * (\frac{1}{2} + Adv_{\mathcal{A}}^{SPP}(\lambda)) \geq \frac{1}{2} + \frac{1}{Q^2}(1 - \frac{1}{Q})^{Q-2} * Adv_{\mathcal{A}}^{SPP}(\lambda) \geq \frac{1}{2} + \frac{1}{4(Q-1)^2} * Adv_{\mathcal{A}}^{SPP}(\lambda)$. Thus, $Adv_{\mathcal{B}}^{TRD1}(\lambda) \geq \frac{1}{4(Q-1)^2} Adv_{\mathcal{A}}^{SPP}(\lambda)$. Therefore, $Adv_{\mathcal{A}}^{SPP}(\lambda) \leq 4(Q-1)^2 * Adv_{\mathcal{B}}^{TRD1}(\lambda)$. □

5 User Key Indistinguishability

In this section, we discuss a different security property of our PEKS scheme. We call it User Key Indistinguishability. We show that our scheme does not allow a probabilistic polynomial time adversary to link a trapdoor to a user. Let us consider the following security experiment $Exp_{\mathcal{A}}^{UKI}(\lambda)$.

Definition 4. *(UKI-Security).*

$Exp_{\mathcal{A}}^{UKI}(\lambda)$
$(G, G_1, p, g, e, H) \leftarrow Setup(\lambda)$
$(Sk_{R0}, Pk_{R0}) \leftarrow KeyGen_R(G, G_1, p, g, e, H)$
$(Sk_{R1}, Pk_{R1}) \leftarrow KeyGen_R(G, G_1, p, g, e, H)$
$(Pk_C, \Pi_C) \leftarrow KeyGen_C(G, G_1, p, g, e, H)$
$(st_1, Pk_S, Pk_T) \leftarrow \mathcal{A}_0^{KeyGen_S}(G, G_1, p, g, e, H, Pk_{R0}, Pk_{R1}, Pk_C, \Pi_C)$
$if\ e(Pk_T, g) \neq e(Pk_C, Pk_S)$
$return\ 0$
$(w, st_2) \leftarrow \mathcal{A}_1^{Trapdoor(\cdot), H(\cdot)}(st_1)$
$d \xleftarrow{\$} \{0, 1\}$
$T_w = Trapdoor(Sk_{Rd}, w, \cdot)$
$d' \leftarrow \mathcal{A}_2^{Trapdoor(\cdot), H(\cdot)}(st_2, T_w)$
$return\ d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$*, against the security experiment* $Exp_{\mathcal{A}}^{UKI}(\lambda)$ *is denoted as* $Adv_{\mathcal{A}}^{UKI}(\lambda)$*. We define it as follows:*

$$Adv_{\mathcal{A}}^{UKI}(\lambda) = \left| Pr[Adv_{\mathcal{A}}^{UKI}(\lambda)] - \frac{1}{2} \right|$$

We say that a PEKS scheme is UKI-secure if for all probabilistic polynomial time adversary $\mathcal{A}$*,* $Adv_{\mathcal{A}}^{UKI}(\lambda)$ *is negligible.*

Theorem 2. *Our PEKS scheme offers UKI-security.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, against the security experiment $Exp_{\mathcal{A}}^{UKI}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$. $\mathcal{B}$ works as follows:

When $\mathcal{B}_0$ is invoked with the inputs $G, G_1, e, g, g^a, g^{b_0}, g^{b_1}, g^c$, $\mathcal{B}_0$ implicitly sets $(Sk_{R0}, Pk_{R_0}) = (b_0, g^{b_0})$, $(Sk_{R1}, Pk_{R1}) = (b_1, g^{b_1})$, and $(Sk_C, Pk_C) = (a, g^a)$. $\mathcal{B}_0$ generates simulated proof Π_C of knowledge of Sk_C, given Pk_C. $\mathcal{B}_0$ invokes $\mathcal{A}_0()$ with $G, G_1, p, g, h, e, H, Pk_{R0}, Pk_{R1}, Pk_C, \Pi_C$. $\mathcal{A}_0$ returns st_1, Pk_S, and Pk_T. $\mathcal{B}_0$ learns about Sk_S from the simulation of $\mathcal{A}_0$. Now, $\mathcal{B}_0$ invokes $\mathcal{A}_1$. $\mathcal{B}_0$ answers all queries of $\mathcal{A}_1$ to the oracles $Trapdoor(\cdot)$, and $H(\cdot)$. Whenever, $Trapdoor(w)$ is queried, a $H(w)$ is automatically made. This $H(w)$ is an indirect query. $\mathcal{A}_1$ can also make direct query to the $H(\cdot)$. when $\mathcal{B}_0$ receives a direct or indirect query to the $H(\cdot)$ oracle, for a keyword w, $\mathcal{B}_0$ samples a random $\alpha \in_R \mathbb{Z}_p$, and computes $\mathbf{R} = g^\alpha$. If $H(\tilde{w}) = g^c$ has been returned for a previous query of the form $H(\tilde{w})$, then $\mathcal{B}_0$ returns $H(w) = \mathbf{R}$. If g^c has never been returned for a previous hash query, then $\mathcal{B}_0$ returns $H(w) = \mathbf{A}$, where;

$$H(w) = \begin{cases} g^c & \text{with probability } 1/Q \\ \mathbf{R} & \text{with probability } 1 - 1/Q \end{cases}$$

Now, we describe how $\mathcal{B}_0$ answers to the queries of $\mathcal{A}_1$ to the $Trapdoor(\cdot)$ oracle. Whenever $\mathcal{B}_0$ receives a query to the trapdoor oracle for a keyword w, and a receiver key $Pk_u \in \{Pk_{R0}, Pk_{R1}\}$, it samples random $\tilde{r}$, and computes $T_1 = e(Pk_u, H(w))^{Sk_S} * e(Pk_S, g^{\tilde{r}}), T_2 = e(Pk_T, g^{\tilde{r}})$. $\mathcal{B}_0$ returns $T_w = (T_1, T_2)$ as the response.

Now, $\mathcal{A}_1$ returns a keyword w^*. If $H(w^*)$ was previously queried and g^c was not returned, $\mathcal{B}_0$ sends instruction to $\mathcal{B}_1$ to abort and return a random bit. This instruct is passed to $\mathcal{B}_1$ via the state variable st_2. If $H(w^*)$ was never queried before, $\mathcal{B}_0$ sets $H(w^*) = g^c$. Now, $\mathcal{B}_0$ returns Pk_S. When $\mathcal{B}_1$ is invoked with the inputs T_2, and T_1, then it invokes $\mathcal{A}_2$ with these inputs. $\mathcal{B}_1$ answers all the queries of $\mathcal{A}_2$ to the trapdoor oracle in the same way as was done by $\mathcal{B}_0$. $\mathcal{B}_1$ answers the hash queries of $\mathcal{A}_2$ by returning randomly sampled elements from G. It is easy to see that if $\mathcal{A}_2$ can identify the challenged input, so can $\mathcal{B}_1$.

Let us calculate the probability of success of $\mathcal{B}$. $\mathcal{B}_1$ aborts and returns a random bit if $\mathcal{A}_0$ becomes unsuccessful in setting $H(w^*) = g^c$. This can happen in one of the two situations:

1) $\mathcal{A}_1$ queries $H(w^*)$, but $\mathcal{B}_0$ fails to return g^c.

2) $\mathcal{A}_1$ does not query $H(w^*)$, but $\mathcal{B}_0$ returns g^c in response to a different trapdoor query by $\mathcal{A}_1$.

Let, P_1, and P_2 denote the probability of occurrence of situation 1, and 2 respectively. Hence, $P_1 = 1 - (1 - \frac{1}{Q})^i \frac{1}{Q}$. Here, i is such that the $i+1$'th hash query correspond to the keyword $H(w^*)$. P_1 is maximised when $i = Q-1$, that is if the last hash query corresponds to the challenge keyword. Similarly, $P_2 = 1 - (1 - \frac{1}{Q})^Q$. So, the probability that $\mathcal{B}_1$ will

have to abort is at most $\max(P_1, P_2) = 1 - (1 - \frac{1}{Q})^{Q-1}\frac{1}{Q}$. So, the least probability that $\mathcal{B}_1$ will not abort is $(1 - \frac{1}{Q})^{Q-1}\frac{1}{Q} \geq \frac{1}{4(Q-1)}$.

Now, $Pr[Exp_{\mathcal{B}}^{TRD1}(\lambda) = 1] \geq Pr[\mathcal{B}_1 \text{ will abort }] * \frac{1}{2} + Pr[\mathcal{B}_1 \text{ will not abort }] * Pr[Exp_{\mathcal{A}}^{UKI}(\lambda) = 1] = \frac{1}{2} + Pr[\mathcal{B}_1 \text{ will not abort }] * Adv_{\mathcal{A}}^{UKI}(\lambda) \geq \frac{1}{2} + \frac{1}{4(Q-1)} * Adv_{\mathcal{A}}^{UKI}(\lambda)$. Thus, $Adv_{\mathcal{B}}^{TRD1}(\lambda) \geq \frac{1}{4(Q-1)} * Adv_{\mathcal{A}}^{UKI}(\lambda)$. That is, $Adv_{\mathcal{A}}^{UKI}(\lambda) \leq 4(Q-1) * Adv_{\mathcal{B}}^{TRD1}(\lambda)$. Hence, the result holds. □

6 Server Key Indistinguishability

In this section, we show that our scheme satisfy another security definition, which we call Server Key Indistinguishability. This security notion aims to assess the invulnerability of a PEKS scheme against an attack that aims to link a trapdoor to a cloud server for which it is generated. That is a PPT adversary will not be able to tell from a trapdoor which server key it corresponds to. For this purpose, we consider an adversary that can compromise the data sender. We underscore the fact that if the data owner cannot breach the server key indistinguishability property of our PEKS scheme, no other adversary can do it. Let us now consider the following security experiment.

Definition 5. *(SKI-security).*

$Exp_{\mathcal{A}}^{SKI}(\lambda)$
$(G, G_1, p, g, e, H) \leftarrow Setup(\lambda)$
$(Sk_R, Pk_R) \leftarrow KeyGen_R(G, G_1, p, g, e, H)$
$(Pk_{C0}, \Pi_{C0}) \leftarrow KeyGen_C(G, G_1, p, g, e, H)$
$(Pk_{C1}, \Pi_{C1}) \leftarrow KeyGen_C(G, G_1, p, g, e, H)$
$(st_1, Pk_{S0}, Pk_{S1}, Pk_{T0}, Pk_{T1}) \leftarrow$
$\mathcal{A}_0^{KeyGen_S}(G, G_1, p, g, e, H, PK_R, Pk_{C0}, Pk_{C1}, \Pi_{C0}, \Pi_{C1})$
$if\ \exists j \in [0,1] : e(Pk_{Tj}, g) \neq e(Pk_{Cj}, Pk_{Sj})$
$return\ 0$
$(w, st_2) \leftarrow \mathcal{A}_1^{Trapdoor(\cdot), H(\cdot)}(st_1)$
$d \xleftarrow{\$} \{0,1\}$
$T_w = Trapdoor(Sk_R, w, Pk_{Sd}, Pk_{Td}\cdot)$
$d' \leftarrow \mathcal{A}_2^{Trapdoor(\cdot), H(\cdot)}(st_2, T_w)$
$return\ d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, against the security experiment is defined as

$$Adv_{\mathcal{A}}^{SKI}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{SKI}(\lambda) = 1] - \frac{1}{2} \right|$$

The PEKS scheme will be secure in terms of Server Key Indistinguishability notion if for all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, $Adv_{\mathcal{A}}^{SKI}(\lambda) \leq negl(\lambda)$.

Assumption 10 *Given $G, G_1, e : G \times G \rightarrow G_1$, let us consider the following security experiment $Exp_{\mathcal{A}}^{TRD3}(\lambda)$.*

$Exp_{\mathcal{A}}^{TRD3}(\lambda)$
$g \xleftarrow{\$} G$
$a \xleftarrow{\$} \mathbb{Z}_p$
$c \xleftarrow{\$} \mathbb{Z}_p,\ f \xleftarrow{\$} \mathbb{Z}_p$
$(k, st) \leftarrow \mathcal{A}_0(G, G_1, e, g, g^a, g^c, g^f)$
$A \xleftarrow{\$} G_1$
$B \xleftarrow{\$} G_1$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow (e(g^k, g^r)^a, e(g^k, g^{r+c\cdot f}))$
$\Omega_1 \leftarrow (A, B)$
$d \xleftarrow{\$} \{0, 1\}$
$d' \leftarrow \mathcal{A}_1(st, \Omega_d)$
return $d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, against the security experiment* $Exp_{\mathcal{A}}^{TRD3}(\lambda)$ *is given by*

$$Adv_{\mathcal{A}}^{TRD3}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1] - \frac{1}{2} \right|$$

For all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*,* $Adv_{\mathcal{A}}^{TRD3}(\lambda) \leq negl(\lambda)$*.*

Lemma 11. *Assumption 2 implies assumption 10.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{TRD3}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B}$, against the security experiment $Exp_{\mathcal{B}}^{BDH}(\lambda)$. $\mathcal{B}$ receives as input g^x, g^y, g^z, and a challenge $\Omega_d \in_R \{e(g,g)^{xyz}, R\}$, where $R \in_R G_1$. $\mathcal{B}$ selects random $c, f \in_R \mathbb{Z}_p$, and invokes $\mathcal{A}_0(G, G_1, e, g, g^x, g^c, g^f)$. $\mathcal{A}_0$ returns $k \in \mathbb{Z}_p$. Now, $\mathcal{B}$ invokes $\mathcal{A}_1$ with inputs Ω_d^k, and $e(g^y, g^z)^k * e(g,g)^{kcf}$. $\mathcal{B}$ returns what $\mathcal{A}_1$ returns. It is easy to see that $Adv_{\mathcal{A}}^{TRD3}(\lambda) \leq Adv_{\mathcal{B}}^{BDH}(\lambda)$. □

Lemma 12. *Given* $G, G_1, e : G \times G \to G_1$*, let us consider the following security experiment* $Exp_{\mathcal{A}}^{TRD4}(\lambda)$*.*

$Exp_{\mathcal{A}}^{TRD4}(\lambda)$
$g \xleftarrow{\$} G$
$a_0 \xleftarrow{\$} \mathbb{Z}_p, a_1 \xleftarrow{\$} \mathbb{Z}_p$
$c \xleftarrow{\$} \mathbb{Z}_p, f \xleftarrow{\$} \mathbb{Z}_p$
$(k_0, k_1, st) \leftarrow \mathcal{A}_0(G, G_1, e, g, g^{a_0}, g^{a_1}, g^c, g^f)$
$r \xleftarrow{\$} \mathbb{Z}_p$
$\Omega_0 \leftarrow (e(g^{k_0}, g^r)^{a_0}, e(g^{k_0}, g^{r+c\cdot f}))$
$\Omega_1 \leftarrow (e(g^{k_0}, g^r)^{a_1}, e(g^{k_0}, g^{r+c\cdot f}))$
$d \xleftarrow{\$} \{0, 1\}$
$d' \leftarrow \mathcal{A}_1(st, \Omega_d)$
return $d = d'$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, against the security experiment* $Exp_{\mathcal{A}}^{TRD4}(\lambda)$ *is given by*

$$Adv_{\mathcal{A}}^{TRD4}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1] - \frac{1}{2} \right|$$

For all PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*,* $Adv_{\mathcal{A}}^{TRD4}(\lambda) \leq negl(\lambda)$*.*

Proof. Let us consider the following security experiment:

$Exp_{\mathcal{A}}^{TRD5}(\lambda)$
$g \xleftarrow{\$} G$
$A \xleftarrow{\$} G_1, B \xleftarrow{\$} G_1$
$C \xleftarrow{\$} G_1, D \xleftarrow{\$} G_1$
$\Omega_0 \leftarrow (A, B)$
$\Omega_1 \leftarrow (C, D)$
$d \xleftarrow{\$} \{0, 1\}$
$d' \leftarrow \mathcal{A}(G, G_1, e, g, \Omega_d)$
return $d = d'$

The advantage of an adversary $\mathcal{A}$, against the security experiment $Exp_{\mathcal{A}}^{TRD5}(\lambda)$ is given by

$$Adv_{\mathcal{A}}^{TRD5}(\lambda) = \left| Pr[Exp_{\mathcal{A}}^{TRD5}(\lambda) = 1] - \frac{1}{2} \right|$$

Now, $Adv_{\mathcal{A}}^{TRD4}(\lambda) + Adv_{\mathcal{A}}^{TRD5}(\lambda) = Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1] + Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1] - 1 = Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1 \big| d = 0] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1 \big| d = 1] + Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD5}(\lambda) = 1 \big| d = 0] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD5}(\lambda) = 1 \big| d = 1] - 1 = (Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1 \big| d = 0] + Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD5}(\lambda) = 1 \big| d = 0]) + (Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD4}(\lambda) = 1 \big| d = 1] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD5}(\lambda) = 1 \big| d = 1]) - 1 \leq \frac{1}{2} * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 0] + \frac{1}{2} * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 1] + \frac{1}{2} * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 0] + \frac{1}{2} * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 1] - 1 = 2 * (Pr[d = 0] * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 0] + Pr[d = 1] * Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1 \big| d = 1] - \frac{1}{2}) = 2 * (Pr[Exp_{\mathcal{A}}^{TRD3}(\lambda) = 1] - \frac{1}{2}) = 2 * Adv_{\mathcal{A}}^{TRD3}(\lambda)$. Hence, $Adv_{\mathcal{A}}^{TRD4}(\lambda) \leq 2 * Adv_{\mathcal{A}}^{TRD3}(\lambda)$. Thus, the result holds. □

Theorem 3. *Our PEKS scheme is secure according to the SKI security notion.*

Proof. We show that if there exists an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, against the security experiment $Exp_{\mathcal{A}}^{SKI}(\lambda)$, it could be used in the construction of another adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ against the security experiment $Exp_{\mathcal{B}}^{TRD4}(\lambda)$. $\mathcal{B}$ works as follows. When $\mathcal{B}_0$ is invoked with $G, G_1, e, g, g^{a_0}, g^{a_1}, g^c, g^d$, it implicitly assigns $(Sk_{Ci}, Pk_{Ci}) = (a_i, g^{a_i})$ for $i = 0, 1$. $\mathcal{B}_0$ also assigns $(Sk_R, Pk_R) = (c, g^c)$. It also computes Π_{C0}, and Π_{C1} using simulation method. Then $\mathcal{B}_0$ invokes $\mathcal{A}_0$ with all the necessary inputs. $\mathcal{A}_0$ returns $st_1, Pk_{S0}, Pk_{S1}, Pk_{T0}$, and Pk_{T_1}. $\mathcal{B}_0$ can find Sk_{S0}, and Sk_{S1} from the simulation of $\mathcal{A}_0$. Now, $\mathcal{B}_0$ invokes $\mathcal{A}_1(st_1)$. $\mathcal{B}_0$ responds to all the queries of $\mathcal{A}_1$ to the oracles $Trapdoor(\cdot)$, and $H(\cdot)$. When $\mathcal{A}_1$ makes a call to the $Trapdoor(\cdot)$ oracle, it specifies the sender's public key (Pk_{S0} or Pk_{S1}), and the corresponding keyword. In answering a trapdoor query, $\mathcal{B}_0$ needs to first make a $H(\cdot)$ query. We call such queries indirect queries to the hash oracle. These indirect queries to the hash oracle are replied in the same way as the direct queries to the hash oracles. Whenever $\mathcal{B}_0$ receives a hash query against a keyword w, $\mathcal{B}_0$ selects a random element $x \in_R \mathbb{Z}_p$, and checks whether there is an entry of the form $\langle w, -, g^d \rangle$ in $listH$. If there is an entry of the

form $\langle w, -, g^d \rangle$ in $listH$, $\mathcal{B}_0$ returns $H(w) = g^x$, and adds $\langle w, x \rangle$ to the list $listH$. If there is no entry of the form $\langle w, -, g^d \rangle$ in $listH$, $\mathcal{B}_0$ returns $H(w) = R$, where

$$R = \begin{cases} g^d & \text{with probability } 1/Q \\ g^x & \text{with probability } 1 - 1/Q \end{cases}$$

Here, Q is the total number of hash queries made by $\mathcal{A}_1$. If $R = g^d$, then $\mathcal{B}_0$ adds $\langle w, -, g^d \langle$ to $listH$. Else if $R = g^x$ then $\mathcal{B}_0$ adds $\langle w, x, g^x \langle$ to $listH$. Whenever a trapdoor query is received for a keyword w, and a sender key Pk_S, and a trapdoor key Pk_T, $\mathcal{B}_0$ first answers to $H(w)$. After this hash query $listH$ will have an entry $\langle w, \alpha, g^x \rangle$, where α is either x or $-$. Then $\mathcal{B}_0$ selects a random $r \in_R \mathbb{Z}_p$, and computes $T_{w1} = e(Pk_R, g^x)^{Sk_S} * e(Pk_S, g^r)$, $T_{w2} = e(Pk_T, g^r)$, and $T_w = (T_{w1}, T_{w2})$. Note that $\mathcal{B}_0$ knows the value of both Sk_{S0}, and Sk_{S1}. So, it can compute T_{w1}. Then $\mathcal{B}_0$ returns T_w as the response for the trapdoor query. Then $\mathcal{A}_1$ returns a keyword $\hat{w}$, and its auxiliary variable st_2. Now, $\mathcal{B}_0$ returns (PK_{S0}, Pk_{S1}, st_2). If $H(\hat{w})$ was queried, and g^c was not returned, then $\mathcal{B}_0$ sends instruction to $\mathcal{B}_1$ to return a random bit. This instruction is sent through the auxiliary variable that is generated by $\mathcal{B}_0$, and is fed to $\mathcal{B}_1$ as an input. When $\mathcal{B}_1$ receives the challenge $\Omega_d = (A_1, A_2)$, $\mathcal{B}_1$ computes $T_1 = A_2$, and $T_2 = A_1$. If $\mathcal{B}_0$ has sent it instruction to return a random bit, then it does so. Otherwise, it invokes $\mathcal{A}_2(st_2, (T_1, T_2))$. $\mathcal{B}_1$ answers all the queries of $\mathcal{A}_2$ to the oracles $Trapdoor(\cdot)$, and $H(\cdot)$ the same way as $\mathcal{B}_0$ did for $\mathcal{A}_1$. Finally, $\mathcal{A}_2$ will return a bit d'. $\mathcal{B}_1$ should return the same bit.

Let us now calculate the success probability of $\mathcal{B}$. $\mathcal{B}_1$ aborts and returns a random bit if $\mathcal{B}_0$ cannot set $H(\hat{w}) = g^c$. Since, $\mathcal{B}_0$ returns g^c with probability $1/Q$ for a random hash query, the probability that $\mathcal{B}_0$ would be able to set $H(\hat{w}) = g^c$ is at least $(1 - \frac{1}{Q})^{Q-1}\frac{1}{Q} \geq \frac{1}{4(Q-1)}$. So, the least probability that $\mathcal{B}_1$ will not abort is $(1 - \frac{1}{Q})^{Q-1}\frac{1}{Q} \geq \frac{1}{4(Q-1)}$.

Now, $Pr[Exp_{\mathcal{B}}^{TRD4}(\lambda) = 1] \geq Pr[\mathcal{B}_1 \text{ will abort }] * \frac{1}{2} + Pr[\mathcal{B}_1 \text{ will not abort }] * Pr[Exp_{\mathcal{A}}^{SKI}(\lambda) = 1] = \frac{1}{2} + Pr[\mathcal{B}_1 \text{ will not abort }] * Adv_{\mathcal{A}}^{SKI}(\lambda) \geq \frac{1}{2} + \frac{1}{4(Q-1)} * Adv_{\mathcal{A}}^{SKI}(\lambda)$. Thus, $Adv_{\mathcal{B}}^{TRD4}(\lambda) \geq \frac{1}{4(Q-1)} * Adv_{\mathcal{A}}^{UKI}(\lambda)$. That is, $Adv_{\mathcal{A}}^{SKI}(\lambda) \leq 4(Q-1) * Adv_{\mathcal{B}}^{TRD4}(\lambda)$. Hence, the result holds. □

7 Performance

In this section, we analyse the performance of our encryption scheme. In Table 1 we present the number of operations needed for each of the component of our scheme and compare the performance of our scheme with that of the PAEKS [14]. In Table 2 we provide comparative study with other searchable schemes.

PAEKS approach of [14]: In [14], authors took 3 exponentiation operations and 1 hashing for encryption which is same in our encryption. For trapdoor generation PAEKS needs one exponentiation one hashing and one mapping whereas we need one exponentiation, one hashing and two mapping. For searching, i.e. Test function, PAEKS need only 2 mapping whereas we need one exponentiation and two mapping(see Table 1). At the expence of these few extra computation we provide UKI security (see Definition 4) and SKI security (see Definition 4).

8 Conclusion

n this paper, we propose a new symmetric searchable encryption scheme that supports phrase search. Our scheme encrypts a sentence of a document separately. The encryption

Operation	#(Exp)	(Mult)	#(Bp)	#(Div)	#(Hash)
Encryption	3	1	0	0	1
Trapdoor	1	1	2	0	1
Test	1	0	2	2	0

Table 1. Complexity of our scheme in terms of number of exponentiation(Exp), Multiplication (Mult) and Bilinear map (Bp), Division (Div) in $\mathbb{G}$ and Inverse (Inv) in $\mathbb{Z}_p$ operations.

Property	SSE [12]	PSS [23]	[25]	LPSSE [17]	Π_{ss} [13]	PAEKS [14]	This paper
Non-adaptive security	yes	yes	no	yes	yes	yes	yes
Adaptive security	yes	no	no	no	no	yes	yes
Security against active adversary	no	no	no	no	yes	yes	yes
Probabilistic trapdoor	no	no	no	no	yes	yes	yes
Client storage	no	dictionary	trusted server	no	no	no	no
No. of rounds	1	2	2	1	1	1	1
Storage cost	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
No. of encryptions per keyword	$O(N)$	$O(N)$	$O(N)$	$O(N)$	1	1	1
Leakage following a search	–	yes	yes	yes	yes	no	no
UKI security	no	no	no	no	no	no	yes
SKI security	no	no	no	no	no	no	yes

Table 2. Properties and performances of different searchable encryption schemes. Search time is per keyword, where N is the number of documents.

of a sentence yields a set of encrypted keywords and an adjacency matrix that stores the ordering information about the keywords in the sentence. To search in a ciphertext corresponding to a sentence, a trapdoor is required. When the server searches in a sentence using a trapdoor, the server does not learn the physical position of the phrase in the encrypted sentence. The proposed scheme avoids the leakage associated with the searchable encryption scheme by previous works. We have proved the security properties of our scheme. We have also implemented the scheme using C and the OpenSSL library. Our theoretical and experimental results show that our scheme is suitable for deployment in real world scenarios.

Acknowledgement : The authors are thankful to the anonymous reviewers, whose comments greatly improved the quality of the manuscript. We also wish to thank Professor Feng Hao and Dr Samiran Bag for providing several useful and valuable suggestions.

References

1. http://www.fon.hum.uva.nl/david/ma_ssp/2007/timit/train/dr5/fsdc0/. 2007.
2. Samiran Bag, Indranil Ghosh Ray, and Feng Hao. A new leakage resilient symmetric searchable encryption scheme for phrase search. In *19th International Conference on Security and Cryptography SECRYPT*, pages 366–373. SciTePress, 2022.
3. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Annual international cryptology conference*, pages 41–55. Springer, 2004.
4. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 506–522, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

5. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption With Keyword Search. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 506–522. Springer, 2004.
6. Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E Skeith III. Public Key Encryption That Allows PIR Queries. In *Annual International Cryptology Conference*, pages 50–67. Springer, 2007.
7. Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-Preserving Multi-Keyword Ranked Search Over Encrypted Cloud Data. volume 25, pages 222–233. IEEE, 2014.
8. David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.
9. David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-Scalable Searchable Symmetric Encryption With Support for Boolean Queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.
10. Melissa Chase and Emily Shen. Substring-searchable symmetric encryption. *Proceedings on Privacy Enhancing Technologies*, 2015(2):263 – 281, 01 Jun. 2015.
11. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 79–88, New York, NY, USA, 2006. Association for Computing Machinery.
12. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. volume 19, pages 895–934. IOS Press, 2011.
13. I. Ghosh Ray, Y. Rahulamathavan, and M. Rajarajan. A new lightweight symmetric searchable encryption scheme for string identification. volume 8, pages 672–684, 2020.
14. Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. volume 403-404, pages 1 – 14, 2017.
15. Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM, 2012.
16. Z. A. Kissel and J. Wang. Verifiable phrase search over encrypted data secure against a semi-honest-but-curious adversary. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 126–131, 2013.
17. Mingchu Li, Wei Jia, Cheng Guo, Weifeng Sun, and Xing Tan. LPSSE: Lightweight Phrase Search With Symmetric Searchable Encryption in Cloud Storage. In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, pages 174–178. IEEE, 2015.
18. M. Naveed, M. Prabhakaran, and C. A. Gunter. Dynamic searchable encryption via blind storage. In *2014 IEEE Symposium on Security and Privacy*, pages 639–654, 2014.
19. Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.
20. Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical Dynamic Searchable Encryption With Small Leakage. In *NDSS*, volume 14, pages 23–26, 2014.
21. Shi-Feng Sun, Joseph K. Liu, Amin Sakzad, Ron Steinfeld, and Tsz Hon Yuen. An efficient non-interactive multi-client searchable encryption with support for boolean queries. In Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, and Catherine Meadows, editors, *Computer Security – ESORICS 2016*, pages 154–172, Cham, 2016. Springer International Publishing.
22. Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y. Thomas Hou, and Hui Li. Privacy-preserving multi-keyword text search in the cloud supporting similaritybased ranking. In *IN ASIACCS 2013*, 2013.
23. Yinqi Tang, Dawu Gu, Ning Ding, and Haining Lu. Phrase Search Over Encrypted Data With Symmetric Encryption Scheme. In *2012 32nd International Conference on Distributed Computing Systems Workshops*, pages 471–480. IEEE, 2012.
24. B. Wang, S. Yu, W. Lou, and Y. T. Hou. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 2112–2120, 2014.
25. S. Zittrower and C. C. Zou. Encrypted phrase searching in the cloud. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 764–770, 2012.