7-2015

A new general framework for secure public key encryption with keyword search

Rongmao CHEN

Yi MU

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Fuchun GUO

Xiaofen WANG

Citation

CHEN, Rongmao; MU, Yi; YANG, Guomin; GUO, Fuchun; and WANG, Xiaofen. A new general framework for secure public key encryption with keyword search. (2015). *Proceedings of the 20th Australasian Conference, Brisbane, Australia, 2015 June 29 - July 1*. 9144, 59-76.
Available at: https://ink.library.smu.edu.sg/sis_research/7372

A New General Framework for Secure Public Key Encryption with Keyword Search

Rongmao Chen[1,2](✉), Yi Mu[1](✉), Guomin Yang[1], Fuchun Guo[1], and Xiaofen Wang[1,3]

[1] Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
{rc517,ymu,gyang,fuchun}@uow.edu.au, wangxuedou@sina.com
[2] College of Computer, National University of Defense Technology, Changsha, China
[3] Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

Abstract. Public Key Encryption with Keyword Search (PEKS), introduced by Boneh et al. in *Eurocrypt'04*, allows users to search encrypted documents on an untrusted server without revealing any information. This notion is very useful in many applications and has attracted a lot of attention by the cryptographic research community. However, one limitation of all the existing PEKS schemes is that they cannot resist the Keyword Guessing Attack (KGA) launched by a malicious server. In this paper, we propose a new PEKS framework named Dual-Server Public Key Encryption with Keyword Search (DS-PEKS). This new framework can withstand all the attacks, including the KGA from the two untrusted servers, as long as they do not collude. We then present a generic construction of DS-PEKS using a new variant of the Smooth Projective Hash Functions (SPHFs), which is of independent interest.

Keywords: Dual-server public key encryption with keyword search · Smooth projective hash function

1 Introduction

To enable encrypted documents searchable on an untrusted server without revealing any information, Boneh et al. [10] introduced the notion of public key encryption with keyword search (PEKS) in Eurocrypt'04. A PEKS system has many potential applications including private databases and data mining where the users are concerned about their data privacy. In a PEKS system, a sender attaches some encrypted keywords (referred to as PEKS ciphertexts) with the encrypted data. The receiver then sends the trapdoor of a to-be-searched keyword to the server for data searching. Given the trapdoor and the PEKS ciphertext, the server can test whether the keyword underlying the PEKS ciphertxt is equal to the one selected by the receiver. If so, the server sends the matching encrypted data to the receiver.

© Springer International Publishing Switzerland 2015
E. Foo and D. Stebila (Eds.): ACISP 2015, LNCS 9144, pp. 59–76, 2015.
DOI: 10.1007/978-3-319-19962-7_4

Inside Keyword Guessing Attack. A PEKS scheme is considered secure if it reveals no information about the data to the server. To be more precise, the server can only locate the message specified by the receiver but cannot learn anything else. Nevertheless, none of the existing PEKS schemes is secure against Keyword Guessing Attack (KGA) launched by a malicious server, which is also known as inside KGA. Specifically, given a trapdoor, the adversarial server can choose a guessing keyword from the keyword space and then use the keyword to generate a PEKS ciphertext. The server then can test whether the guessing keyword is the one underlying the trapdoor. This *guessing-then-testing* procedure can be repeated until the correct keyword is found. Such a guessing attack has also been considered in many password-based systems. However, the attack can be launched more efficiently against PEKS since the keyword space is roughly the same as a normal dictionary (e.g., all the meaningful English words), which has a much smaller size than a password dictionary (e.g., all the words containing 6 alphanumeric characters).

Unfortunately, we see that the inside KGA is an inherent vulnerability of the existing PEKS framework which only involves one server. This is essentially due to the following facts.

- **Public Generation of PEKS Ciphertexts**. The generation of a PEKS ciphertext only involves public information; hence the server can generate the PEKS ciphertext for any keyword.
- **Stand-alone Testing.** The server can independently check whether a PEKS ciphertext is consistent with a trapdoor from the receiver.
- **Correctness/consistency of PEKS.** The correctness required by PEKS ensures that the test function always outputs the correct answer.

It is obvious that we should not sacrifice the correctness of PEKS; otherwise we will essentially lose the search functionality. One possible solution to overcome the problem is to disable the public generation of PEKS ciphertexts, which means the PEKS encryption performed by a sender should also take as input some secret information only known by the sender (e.g., the sender's private key). However, this solution is not practical either, since the receiver will need to embed some information of the sender (e.g., the public key) in the generation of a trapdoor. But the receiver may not be aware of the identity of the sender when performing the search, and multiple trapdoors for a keyword need to be generated for all the potential senders in the system.

Our Contributions. The contributions of this paper are two-fold. Firstly, based on the above observations, we propose to overcome the inside KGA by disallowing the stand-alone testing performed by a server. Specifically, we formally define a new PEKS framework named *Dual-Server Public Key Encryption with Keyword Search* (DS-PEKS). Secondly, we show a generic construction of DS-PEKS using a newly defined variant of *Smooth Projective Hash Function* (SPHF) which maybe of independent interest.

Overview of Techniques. The key idea of the new framework is to split the testing functionality of the PEKS system into two parts which are handled by two independent servers: *front server* and *back server*. Upon receiving a query from the receiver, the front server pre-processes the trapdoor and all the PEKS ciphertexts using its private key, and then sends some *internal testing-states* to the back server with the corresponding trapdoor and PEKS ciphertexts hidden. The back server can then decide which documents are queried by the receiver using its private key and the received internal testing-states from the front server. We show that under such a setting neither the front server nor the back server can launch the keyword guessing attack. It is worth noting that in the above process, we require that the two servers do not collude since otherwise it goes back to the one-server setting in which inside KGA cannot be prevented.

Smooth Projective Hash Function (SPHF) is originally introduced by by Cramer and Shoup [13] for construction of CCA-secure public key encryption schemes. Roughly speaking, an SPHF can be defined based on a domain $\mathcal{X}$ and an $\mathcal{NP}$ language $\mathcal{L}$, where $\mathcal{L} \subset \mathcal{X}$. The key property of SPHF is that the projection key uniquely determines the hash value of any word in the language $\mathcal{L}$ (*projective*) but gives almost no information about the hash value of any point in $\mathcal{X} \setminus \mathcal{L}$ (*smooth*). In our paper, the SPHF used is based on the hard-on-the-average $\mathcal{NP}$-language and hence is also *pseudo-random* [15]. That is, given a word $W \in \mathcal{L}$, without the corresponding witness and the secret hashing key, the distribution of its hash value is computationally indistinguishable from a uniform distribution. Our newly defined variant, named Lin-Hom SPHF, additionally requires the underlying language and the hash function to be *linear* and *homomorphic*. In a generic DS-PEKS construction, to generate the PEKS ciphertext of a keyword, the sender picks a word randomly from $\mathcal{L}$ and computes its two hash values using the witness and the public key (projection key) of the front and back servers respectively. The keyword is then concealed with these two hash values in the PEKS ciphertext. One can see that due to the *pseudo-randomness* of the SPHF, given a PEKS ciphertext, the two servers cannot learn any information about the underlying keyword individually as they do not know the witness or the private key (secret hashing key) of each other. The receiver generates the trapdoor in the same way and hence security of the trapdoor can also be guaranteed. In the pre-processing stage, the front server first removes the pseudo-random hash values in the trapdoor and the PEKS ciphertext for the back server using its private key. Due to the *linear* and *homomorphic* properties of Lin-Hom SPHF, the front server can re-randomise the *internal testing-state* to preserve the keyword privacy from the back server who can only determine whether the two keywords underlying the *internal testing-state* are the same or not. We can see that in this way, the security of DS-PEKS against inside keyword guessing attack can be obtained.

1.1 Related Work

Following Boneh et al.'s seminal work [10], Abdalla et al. [1] formalized anonymous IBE (AIBE) and presented a generic construction of searchable encryption from AIBE. They also showed how to transfer a hierarchical IBE (HIBE) scheme into a public key encryption with temporary keyword search (PETKS) where the trapdoor is only valid in a specific time interval. Waters [22] showed that the PEKS schemes based on bilinear map could be applied to build encrypted and searchable auditing logs. In order to construct a PEKS secure in the standard model, Khader [19] proposed a scheme based on the k-resilient IBE and also gave a construction supporting multiple-keyword search. The first PEKS scheme without pairings was introduced by Crescenzo and Saraswat [14]. The construction is derived from Cock's IBE scheme [12] which is not very practical.

Byun et al. [11] introduced the off-line keyword guessing attack against PEKS as keywords are chosen from a much smaller space than passwords and users usually use well-known keywords for searching documents. They also pointed out that the scheme proposed in Boneh et al. [10] was susceptible to keyword guessing attack. Inspired by the work of Byun *et al.* [11], Yau et al. [23] demonstrated that outside adversaries that capture the trapdoors sent in a public channel can reveal the encrypted keywords through off-line keyword guessing attacks and they also showed off-line keyword guessing attacks against the (SCF-)PEKS schemes in [4,5]. The first PEKS scheme secure against outside keyword guessing attacks was proposed by Rhee et al. [21]. In [20], the notion of trapdoor indistinguishability was proposed and the authors showed that trapdoor indistinguishability is a sufficient condition for preventing outside keyword-guessing attacks.

Nevertheless, all the schemes mentioned above are found to be vulnerable to keyword guessing attacks from a malicious server (i.e., inside keyword guessing attacks). Jeong et al. [17] showed a negative result that the consistency/correctness of PEKS implies insecurity to inside keyword guessing attacks in PEKS. Their result indicates that constructing secure and consistent PEKS schemes against inside keyword guessing attacks is impossible under the original framework.

1.2 Organization

In Section 2, we propose the new framework, namely DS-PEKS, and present its formal definition and security models. We then define a new variant of smooth projective hash function (SPHF), which satisfies the linear and homomorphic properties, for a special class of NP languages in Section 3. A generic construction of DS-PEKS from LH-SPHF is shown in Section 4 with formal correctness analysis and security proofs. We then give a conclusion in Section 5.

2 Dual-Server Public Key Encryption with Keyword Search

In this section, we formally define the Dual-Server Public Key Encryption with Keyword Search (DS-PEKS) and its security model.

2.1 Overview of DS-PEKS

As introduced in [10], a traditional PEKS scheme is defined by a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$. The KeyGen algorithm is used for the generation of public/private key pair for the receiver. The sender runs PEKS to generate keyword ciphertexts which are attached to the data. A receiver then uses Trapdoor to generate the trapdoor for any keyword he/she wants the server to search for. Given the trapdoors, the server can run the algorithm Test to locate the data containing the keywords specified by the receiver. As shown previously, it is impossible to construct a PEKS scheme secure against keyword guessing attack from the malicious server if we follow the traditional PEKS framework due to the public generation of PEKS ciphertexts, stand-alone testing, and the correctness of PEKS. In our proposed framework, namely DS-PEKS, we disallow the stand-alone testing to obtain the security against inside keyword guessing attacks. Roughly speaking, DS-PEKS consists of (KeyGen, DS-PEKS, DS-Trapdoor, FrontTest, BackTest). To be more precise, the KeyGen algorithm generates the public/private key pairs of the front and back servers instead of that of the receiver. Moreover, the trapdoor generation algorithm DS-Trapdoor defined here is public while in the traditional PEKS definition [5,10], the algorithm Trapdoor takes as input the receiver's private key. Such a difference is due to the different structures used by the two systems. In the traditional PEKS, since there is only one server, if the trapdoor generation algorithm is public, then the server can launch an off-line guessing attack against a keyword ciphertext to recover the encrypted keyword. As a result, it is impossible to achieve the semantic security as defined in [5,10]. However, as we will show later, under the DS-PEKS framework, we can still achieve semantic security when the trapdoor generation algorithm is public. Another difference between the traditional PEKS and our proposed DS-PEKS is that the test algorithm is divided into two algorithms, FrontTest and BackTest run by two independent servers. This is essential for achieving security against the inside keyword guessing attacks.

2.2 Definition

Syntax. A DS-PEKS scheme is defined by the following algorithms.

- Setup(1^λ). Takes as input the security parameter λ, generates the system parameters P;
- KeyGen(P). Takes as input the systems parameters P, outputs the public/secret key pairs (pk_{FS}, sk_{FS}), and (pk_{BS}, sk_{BS}) for the front server, and the back server respectively;
- DS-PEKS$(P, pk_{FS}, pk_{BS}, kw_1)$. Takes as input P, the front server's public key pk_{FS}, the back server's public key pk_{BS} and the keyword kw_1, outputs the PEKS ciphertext CT_{kw_1} of kw_1;
- DS-Trapdoor$(P, pk_{FS}, pk_{BS}, kw_2)$. Takes as input P, the front server's public key pk_{FS}, the back server's public key pk_{BS} and the keyword kw_2, outputs the trapdoor T_{kw_2};
- FrontTest$(P, sk_{FS}, CT_{kw_1}, T_{kw_2})$. Takes as input P, the front server's secret key sk_{FS}, the PEKS ciphertext CT_{kw_1} and the trapdoor T_{kw_2}, outputs the internal testing-state C_{ITS};

– BackTest(P, sk_{BS}, C_{ITS}). Takes as input P, the back server's secret key sk_{BS} and the internal testing-state C_{ITS}, outputs the testing result 0 or 1;

Correctness. It is required that for any keyword kw_1, kw_2, and $CT_{kw_1} \leftarrow$ DS-PEKS$(P, pk_{FS}, pk_{BS}, kw_1)$, $T_{kw_2} \leftarrow$ DS-Trapdoor$(P, pk_{FS}, pk_{BS}, kw_2)$, we have that BackTest$(P, sk_{BS}, C_{ITS}) = 1$ if $kw_1 = kw_2$, otherwise 0.

2.3 Security Models

In this subsection, we formalise the security models for DS-PEKS. We define the following security models for a DS-PEKS scheme against the adversarial front and back servers, respectively. We should note that the following security models also imply the security guarantees against the outside adversaries which have less capability compared to the servers.

Adversarial Front Server. In this part, we define the security against an adversarial front server. Precisely, we introduce two games, namely semantic-security against chosen keyword attack and indistinguishability against keyword guessing attack[1] to capture the security of PEKS ciphertext and trapdoor, respectively.

Semantic-Security against Chosen Keyword Attack (SS-CKA). In the following, we define the semantic-security against chosen keyword attack which guarantees that no adversary is able to distinguish a keyword from another one given the corresponding PEKS ciphertext. That is, the PEKS ciphertext does not reveal any information about the underlying keyword to any adversary.

– Setup. The challenger runs the KeyGen(λ) algorithm to generate key pairs (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}). It gives $(pk_{FS}, sk_{FS}, pk_{BS})$ to the attacker;
– Test query-I. The attacker can adaptively make the test query for any keyword and any PEKS ciphertext of its choice. The challenger returns 1 or 0 as the test result to the attacker;
– Challenge. The attacker sends the challenger two keywords kw_0, kw_1. The challenger picks $b \xleftarrow{\$} \{0, 1\}$ and generates

$$CT^*_{kw} \leftarrow \mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_b).$$

The challenger then sends CT^*_{kw} to the attacker;
– Test query-II. The attacker can continue the test query for any keyword and any PEKS ciphertext of its choice except of the challenge keywords kw_0, kw_1. The challenger returns 1 or 0 as the test result to the attacker;
– Output. Finally, the attacker outputs its guess $b' \in \{0, 1\}$ on b and wins the game if $b = b'$.

[1] In this paper, we use two different terms, namely semantic security and indistinguishability, to define the security for the keyword ciphertext and the trapdoor, respectively. However, as for normal public key encryption, these two terms are equivalent.

We refer to such an adversarial front server $\mathcal{A}$ in the above game as an SS-CKA adversary and define its advantage as

$$\mathsf{Adv}_{\mathcal{FS},\mathcal{A}}^{\mathsf{SS\text{-}CKA}}(\lambda) = \Pr[b = b'] - 1/2.$$

Indistinguishability against Keyword Guessing Attack (IND-KGA). This model captures that the trapdoor reveals no information about the underlying keyword to the adversarial front server. We define the security model as follows.

- Setup. The challenger runs the $\mathsf{KeyGen}(\lambda)$ algorithm to generate key pairs (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}). It gives $(pk_{FS}, sk_{FS}, pk_{BS})$ to the attacker;
- Test query-I. The attacker can adaptively make the test query for any keyword and any PEKS ciphertext of its choice. The challenger returns 1 or 0 as the test result to the attacker;
- Challenge. The attacker sends the challenger two keywords kw_0, kw_1. The challenger picks $b \xleftarrow{\$} \{0,1\}$ and generates

$$T_{kw}^{*} \leftarrow \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw_b).$$

  The challenger then sends T_{kw}^{*} to the attacker;
- Test query-II. The attacker can continue issue the test query for any keyword and any PEKS ciphertext of its choice except of the challenge keywords kw_0, kw_1. The challenger returns 1 or 0 as the test result to the attacker;
- Output. Finally, the attacker outputs its guess $b' \in \{0,1\}$ on b and wins the game if $b = b'$.

We refer to such an adversarial front server $\mathcal{A}$ in the above game as an IND-KGA adversary and define its advantage as

$$\mathsf{Adv}_{\mathcal{FS},\mathcal{A}}^{\mathsf{IND\text{-}KGA}}(\lambda) = \Pr[b = b'] - 1/2.$$

Adversarial Back Server. The security models of SS-CKA and IND-KGA in terms of an adversarial back server are similar to those against an adversarial front server.

Semantic-Security against Chosen Keyword Attack. Here the game against an adversarial back server is the same as the one against an adversarial front server except that the adversary is given the private key of the back server instead of that of the front server. We omit the details here for simplicity.

We refer to the adversarial back server $\mathcal{A}$ in the SS-CKA game as an SS-CKA adversary and define its advantage as $\mathsf{Adv}_{\mathcal{BS},\mathcal{A}}^{\mathsf{SS\text{-}CKA}}(\lambda) = \Pr[b = b'] - 1/2$.

Indistinguishability against Keyword Guessing Attack. Similarly, this security model aims to capture that the trapdoor does not reveal any information to the back server and hence is the same as that against the front server except that the adversary owns the private key of the back server instead of that of the front server. Therefore, we also omit the details here.

We refer to the adversarial back server $\mathcal{A}$ in the IND-KGA game as an IND-KGA adversary and define its advantage as $\mathsf{Adv}_{\mathcal{BS},\mathcal{A}}^{\mathsf{IND\text{-}KGA}}(\lambda) = \Pr[b = b'] - 1/2$.

Indistinguishability against Keyword Guessing Attack-II (IND-KGA-II). Apart from the above two security models, we should also guarantee that the internal testing-state does not reveal any information about the keyword to the back server. We hence define another type of keyword guessing attack to capture such a requirement. The security, namely Indistinguishability against Keyword Guessing Attack-II guarantees that the back server cannot learn any information about the keywords from the internal testing-state. The security model is defined as follows.

- Setup. The challenger runs the $\mathsf{KeyGen}(\lambda)$ algorithm to generates key pairs (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}). It gives $(pk_{FS}, pk_{BS}, sk_{BS})$ to the attacker.
- Challenge. The attacker sends the challenger three different keywords $kw_0, kw_1,$
  kw_2. The challenger picks $\{b_1, b_2\} \subset \{0, 1, 2\}$ randomly and computes

  $$\begin{aligned} CT^*_{kw} &\leftarrow \mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_{b_1}), \\ T^*_{kw} &\leftarrow \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw_{b_2}), \\ C^*_{ITS} &\leftarrow \mathsf{FrontTest}(P, sk_{FS}, CT^*_{kw}, T^*_{kw}). \end{aligned}$$

  The challenger then sends C^*_{ITS} to the attacker.
- Output. Finally, the attacker outputs its guess on $\{b_1, b_2\}$ as $\{b'_1, b'_2\} \subset \{0, 1, 2\}$ and wins the game if $\{b'_1, b'_2\} = \{b_1, b_2\}$.

We refer to such an adversary $\mathcal{A}$ in the above two games as a IND-KGA-II adversary and define its advantage as,

$$\mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}}(\lambda) = \Pr[\{b'_1, b'_2\} = \{b_1, b_2\}] - 1/3.$$

We should remark that in the above game, b_1 and b_2 can be equivalent. In this case, the adversary (i.e., back server) will know that the same keyword has been used in the generation of the PEKS ciphertext and the trapdoor, and the adversary's goal is to guess which keyword among the three has been used.

Based on the security models defined above, we give the following security definition for a DS-PEKS scheme.

Definition 1. We say that a DS-PEKS is secure if for any polynomial time attacker $\mathcal{A}_i$ $(i = 1, \ldots, 5)$, we have that $\mathsf{Adv}^{\mathsf{SS\text{-}CKA}}_{\mathcal{BS},\mathcal{A}_1}(\lambda)$,$\mathsf{Adv}^{\mathsf{SS-CKA}}_{\mathcal{BS},\mathcal{A}_2}(\lambda)$, $\mathsf{Adv}^{\mathsf{IND\text{-}KGA}}_{\mathcal{FS},\mathcal{A}_3}(\lambda)$, $\mathsf{Adv}^{\mathsf{IND\text{-}KGA}}_{\mathcal{BS},\mathcal{A}_4}(\lambda)$ and $\mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}_5}(\lambda)$ are all negligible functions of the security parameter λ.

3 Smooth Projective Hash Functions

A central element of our construction for dual-server public key encryption with keyword search is *smooth projective hash function* (SPHF), a notion introduced by Cramer and Shoup [13] and extended for construction of many cryptographic primitives [2,8,15,16,18]. We start with the original definition of an SPHF.

3.1 SPHF

Syntax. An SPHF can be defined based on a domain $\mathcal{X}$ and an $\mathcal{NP}$ language $\mathcal{L}$, where $\mathcal{L}$ contains a subset of the elements of the domain $\mathcal{X}$, i.e., $\mathcal{L} \subset \mathcal{X}$. An SPHF system over a language $\mathcal{L} \subset \mathcal{X}$, onto a set $\mathcal{Y}$, is defined by the following five algorithms $(\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$:

- $\mathsf{SPHFSetup}(1^\lambda)$: generates the global parameters param and the description of an $\mathcal{NP}$ language instance $\mathcal{L}$;
- $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$: generates a hashing key hk for $\mathcal{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}))$: derives the projection key hp from the hashing key hk;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$: outputs the hash value $\mathsf{hv} \in \mathcal{Y}$ for the word W from the hashing key hk;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$: outputs the hash value $\mathsf{hv}' \in \mathcal{Y}$ for the word W from the projection key hp and the witness w for the fact that $W \in \mathcal{L}$.

Property. A smooth projective hash function $\mathcal{SPHF}$ should satisfy the following properties,

- *Correctness.* If a word $W \in \mathcal{L}$ with w the witness, then for all hashing key hk and projection key hp, we have

$$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w).$$

- *Smoothness.* Let a point W be not in the language, i.e., $W \in \mathcal{X} \backslash \mathcal{L}$. Then the following two distributions are statistically indistinguishable :

$$\mathcal{V}_1 = \{(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, \mathsf{hv}) | \mathsf{hv} = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)\},$$

$$\mathcal{V}_2 = \{(\mathcal{L}, \mathsf{param}, W, \mathsf{hp}, \mathsf{hv}) | \mathsf{hv} \overset{\$}{\leftarrow} \mathcal{Y}\},$$

  To be more precise, the quantity of $\sum_{v \in \mathcal{Y}} | \Pr_{\mathcal{V}_1}[\mathsf{hv} = v] - \Pr_{\mathcal{V}_2}[\mathsf{hv} = v]|$ is negligible.

In this paper, we require another important property of smooth projective hash functions that was introduced in [15].

- *Pseudo-Randomness.* If a word $W \in \mathcal{L}$, then without the corresponding witness w, the distribution of the hash output is computationally indistinguishable from a uniform distribution in the view of any polynomial-time adversary $\mathcal{A}$. That is, $\mathcal{A}$ can only win in the PR-game defined below with negligible probability.
  - Setup. The challenger runs $\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}$, generates param,
    $\mathsf{hk}, \mathsf{hp}, \mathcal{L}$ and sends the adversary $(\mathsf{param}, \mathcal{L}, \mathsf{hp})$.
  - $\mathsf{Challenge}$. The challenger picks $W \overset{\$}{\leftarrow} \mathcal{L}, b \overset{\$}{\leftarrow} \{0,1\}$ and generates

$$\mathsf{hv} = \begin{cases} \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W) & b = 1, \\ v \overset{\$}{\leftarrow} \mathcal{Y} & b = 0. \end{cases}$$

    The challenger then sends (W, hv) to $\mathcal{A}$.

- Output. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ on b and wins the game if $b = b'$.

We define the advantage of $\mathcal{A}$ in the above game as,

$$\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda) = \Pr[b = b'] - 1/2.$$

We say $\mathcal{SPHF}$ is *pseudo-random* if for any polynomial time adversary $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda)$ is a negligible function.

3.2 Lin-Hom SPHF

In this paper, we consider a new variant of smooth projective hash function. We consider two new properties: linear and homomorphic, which are defined below. It is worth noting that Abdalla et al. [3] introduced conjunction and disjunction of languages for smooth projective hashing that were later used in the construction of blind signature [7,9], oblivious signature-based envelops [9], and authenticated key exchange protocols for algebraic languages [6]. As shown in the following, our definition for the new SPHF here is different from their work since we consider the operations on the words belonging to the same language, whereas theirs considers operations among different languages.

Let $\mathcal{SPHF}$=(SPHFSetup, HashKG, ProjKG, Hash, ProjHash) be a smooth projective hash function over the language $\mathcal{L} \subset \mathcal{X}$ onto the set $\mathcal{Y}$ and $\mathcal{W}$ be the witness space of $\mathcal{L}$. We first describe the operations on the sets $< \mathcal{L}, \mathcal{Y}, \mathcal{W} >$ as follows.

- $\circledcirc : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$. For any $W_1 \in \mathcal{L}, W_2 \in \mathcal{L}$, $W_1 \circledcirc W_2 \in \mathcal{L}$;
- $\circledast : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathcal{Y}$. For any $y_1 \in \mathcal{Y}, y_2 \in \mathcal{Y}$, $y_1 \circledast y_2 \in \mathcal{Y}$;
- $\odot, \oplus : \mathcal{W} \times \mathcal{W} \rightarrow \mathcal{W}$. For any $w_1 \in \mathcal{W}, w_2 \in \mathcal{W}$, $w_1 \odot w_2 \in \mathcal{W}$ and $w_1 \oplus w_2 \in \mathcal{W}$;
- $\otimes : \mathcal{W} \times \mathcal{L} \rightarrow \mathcal{L}$. For any $w \in \mathcal{W}, W \in \mathcal{L}$, $w \otimes W \in \mathcal{L}$;
- $\bullet : \mathcal{W} \times \mathcal{Y} \rightarrow \mathcal{Y}$. For any $w \in \mathcal{W}, y \in \mathcal{Y}$, $w \bullet y \in \mathcal{Y}$.

Moreover, for any element $y \in \mathcal{Y}$, we define $y \circledast y^{-1} = 1_{\mathcal{Y}}$ which is the identity element of $\mathcal{Y}$.

Our new SPHF requires the underlying language to be also *linear and homomorphic language*, which is defined below.

Definition 2 (Linear and Homomorphic Language). *A language $\mathcal{L}$ is linear and homomorphic if it satisfies the following properties.*

- *For any word $W \in \mathcal{L}$ with witness w and $\Delta w \in \mathcal{W}$, there exists a word $W^* \in \mathcal{L}$ such that $\Delta w \otimes W = W^*$ with the witness $w^* = \Delta w \odot w$.*
- *For any two words $W_1, W_2 \in \mathcal{L}$ with the witness $w_1, w_2 \in \mathcal{W}$ respectively, there exists a word $W^* \in \mathcal{L}$ such that $W_1 \circledcirc W_2 = W^*$ with the witness $w^* = w_1 \oplus w_2$.*

We then give the definition of *Lin-Hom SPHF* as follows.

Definition 3 (Lin-Hom SPHF (LH-SPHF)). *We say $\mathcal{SPHF}$ is a Lin-Hom SPHF (LH-SPHF) if the underlying language $\mathcal{L}$ is a linear and homomorphic language and $\mathcal{SPHF}$ satisfies the following properties.*

– *For any word $W \in \mathcal{L}$ with the witness $w \in \mathcal{W}$ and $\Delta w \in \mathcal{W}$, we have*

$$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), \Delta w \otimes W) = \Delta w \bullet \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W).$$

In other words, suppose $\Delta w \otimes W = W^*$, we have,

$$\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W^*, w^*) = \Delta w \bullet \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w),$$

where $w^* = \Delta w \odot w$.

– *For any two words $W_1, W_2 \in \mathcal{L}$ with the witness $w_1, w_2 \in \mathcal{W}$, we have*

$$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_1 \circledcirc W_2) = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_1) \circledast \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_2).$$

In other words, suppose $W_1 \circledcirc W_2 = W^*$, we have,

$$\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W^*, w^*) = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W_1, w_1) \circledast \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W_2, w_2)$$

where $w^* = w_1 \oplus w_2$.

In this paper, we also assume that the LH-SPHF has the following property: for any $y \in \mathcal{Y}$, $W \in \mathcal{L}$ and the witness $w \in \mathcal{W}$ of W, there exists a projection key hp such that $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w) = y$.

4 Generic Construction of DS-PEKS Using LH-SPHF

In this section, we show how to generically construct a Dual-Server Public Key Encryption with keyword search based on *Lin-Hom Smooth Projective Hash Functions.*

4.1 Generic Construction

Suppose $\mathcal{SPHF} = (\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ is an LH-SPHF over the language $\mathcal{L}$ onto the set $\mathcal{Y}$. Let $\mathcal{W}$ be the witness space of the language $\mathcal{L}$ and $\mathcal{KW}$ be the keyword space. Our generic construction $\mathcal{DS}$-$\mathcal{PEKS}$ works as follows.

$\mathsf{Setup}(1^\lambda)$. Take as input the security parameter λ, run $\mathsf{SPHFSetup}$ algorithm and generate the global parameters param, the description of the language $\mathcal{L}$ and a collision-resistant hash function $\Gamma : \mathcal{KW} \to \mathcal{Y}$. Set the system parameter $P =< \mathsf{param}, \mathcal{L}, \Gamma >$.

$\mathsf{KeyGen}(P)$. Take as input P, run the algorithms $< \mathsf{HashKG}, \mathsf{ProjHash} >$ to generate the public/private key pairs (pk_{FS}, sk_{FS}), (pk_{BS}, sk_{BS}) for the front server and the back server respectively.

$$pk_{FS} \leftarrow \mathsf{HashKG}(P), sk_{FS} = \mathsf{ProjKG}(P, pk_{FS}),$$

$$pk_{BS} \leftarrow \mathsf{HashKG}(P), sk_{BS} = \mathsf{ProjKG}(P, pk_{BS}).$$

$\mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_1)$. Take as input P, pk_{FS}, pk_{BS} and the keyword kw_1, pick a word $W_1 \in \mathcal{L}$ randomly with the witness w_1 and generate the PEKS ciphertext CT_{kw_1} of kw_1 as following.

$$x_1 = \mathsf{ProjHash}(P, pk_{FS}, W_1, w_1),$$

$$y_1 = \mathsf{ProjHash}(P, pk_{BS}, W_1, w_1),$$

$$C_1 = x_1 \circledast y_1 \circledast \Gamma(kw_1).$$

Set $CT_{kw_1} =< W_1, C_1 >$ and return CT_{kw_1} as the keyword ciphertext.

$\mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw_2)$. Take as input P, pk_{FS}, pk_{BS} and the keyword kw_2, pick a word $W_2 \in \mathcal{L}$ randomly with the witness w_2 and generate the trapdoor T_{kw_2} of kw_2 as follows.

$$x_2 = \mathsf{ProjHash}(P, pk_{FS}, W_2, w_2),$$

$$y_2 = \mathsf{ProjHash}(P, pk_{BS}, W_2, w_2),$$

$$C_2 = x_2 \circledast y_2 \circledast \Gamma(kw_2)^{-1}.$$

Set $T_{kw_2} =< W_2, C_2 >$ and return T_{kw_2} as the trapdoor.

$\mathsf{FrontTest}(P, sk_{FS}, CT_{kw}, T_{kw})$. Takes as input P, the front server's secret key sk_{FS}, the PEKS ciphertext $CT_{kw_1} =< W_1, C_1 >$ and the trapdoor $T_{kw_2} =< W_2, C_2 >$, pick $\Delta w \in \mathcal{W}$ randomly, generate the internal testing-state C_{ITS} as follows.

$$W = W_1 \circledcirc W_2,$$

$$x = \mathsf{Hash}(P, sk_{FS}, W),$$

$$C = C_1 \circledast C_2 \circledast x^{-1},$$

$$W^* = \Delta w \otimes W, C^* = \Delta w \bullet C.$$

Set $C_{ITS} =< W^*, C^* >$ and return C_{ITS} as the internal testing-state.

$\mathsf{BackTest}(P, sk_{BS}, C_{ITS})$. Takes as input P, the back server's secret key sk_{BS} and the internal testing-state $C_{ITS} =< W^*, C^* >$, test as follows.

$$\mathsf{Hash}(P, sk_{BS}, W^*) \overset{?}{=} C^*$$

If yes output 1, else output 0.

Correctness Analysis. One can see that the correctness of this construction is guaranteed by the important properties of the LH-SPHF. To be more precise, we give the analysis as follows.

For the algorithm $\mathsf{FrontTest}$, we have

$$\begin{aligned} x &= \mathsf{Hash}(P, sk_{FS}, W) \\ &= \mathsf{Hash}(P, sk_{FS}, W_1 \odot W_2) \\ &= \mathsf{Hash}(P, sk_{FS}, W_1) \circledast \mathsf{Hash}(P, sk_{FS}, W_2) \\ &= \mathsf{ProjHash}(P, pk_{FS}, W_1, w_1) \circledast \mathsf{ProjHash}(P, pk_{FS}, W_2, w_2) \\ &= x_1 \circledast x_2. \end{aligned}$$

Therefore,

$$\begin{aligned} C &= C_1 \circledast C_2 \circledast x^{-1} \\ &= x_1 \circledast y_1 \circledast \mathit{\Gamma}(kw_1) \circledast x_2 \circledast y_2 \circledast \mathit{\Gamma}(kw_2)^{-1} \circledast (x_1 \circledast x_2)^{-1} \\ &= y_1 \circledast y_2 \circledast \mathit{\Gamma}(kw_1) \circledast \mathit{\Gamma}(kw_2)^{-1}. \end{aligned}$$

For the algorithm $\mathsf{BackTest}$, we have

$$\begin{aligned} &\mathsf{Hash}(P, sk_{BS}, W^*) \\ =& \mathsf{Hash}(P, sk_{BS}, \mathit{\Delta} w \otimes W) \\ =& \mathit{\Delta} w \bullet \mathsf{Hash}(P, sk_{BS}, W) \\ =& \mathit{\Delta} w \bullet \mathsf{Hash}(P, sk_{BS}, W_1 \circledcirc W_2). \\ =& \mathit{\Delta} w \bullet (\mathsf{Hash}(P, sk_{BS}, W_1) \circledast \mathsf{Hash}(P, sk_{BS}, W_2)) \\ =& \mathit{\Delta} w \bullet (\mathsf{ProjHash}(P, pk_{BS}, W_1, w_1) \circledast \mathsf{ProjHash}(P, pk_{BS}, W_2, w_2)) \\ =& \mathit{\Delta} w \bullet (y_1 \circledast y_2). \end{aligned}$$

It is easy to see that if $kw_1 = kw_2$, then $\mathsf{Hash}(P, sk_{BS}, W^*) = \mathit{\Delta} w \bullet C = C^*$. Otherwise, $\mathsf{Hash}(P, sk_{BS}, W^*) \neq C^*$ due to the collision-resistant property of the hash function $\mathit{\Gamma}$.

4.2 Security of $\mathcal{DS}$-$\mathcal{PEKS}$

In this subsection, we analyse the security of the above generic construction.

Theorem 1. *The generic construction $\mathcal{DS}$-$\mathcal{PEKS}$ is semantically secure under chosen keyword attacks.*

The above theorem can be obtained from the following two lemmas.

Lemma 1. *For any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SS\text{-}CKA}}_{\mathcal{FS},\mathcal{A}}(\lambda)$ is a negligible function.*

Proof. We define a sequence of games as follows.

$\mathsf{Game0}$. This is the original SS-CKA game against the adversarial front server.

– Setup. The challenger runs the $\mathsf{Setup}, \mathsf{KeyGen}$ to generate system parameter P, key pairs (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}). It then gives adversary $\mathcal{A}$ the key pair $(P, pk_{FS}, sk_{FS}, pk_{BS})$.

– Test Query-I. The adversary makes a query on $< kw, CT >$. Suppose $CT = (W, C)$, the challenger computes the following.

$$T \leftarrow \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw),$$

$$C_{ITS} \leftarrow \mathsf{FrontTest}(P, sk_{FS}, C, T).$$

The challenger then runs the algorithm $\mathsf{BackTest}(P, sk_{BS}, C_{ITS})$ and returns the output to the adversary.

– Challenge. $\mathcal{A}$ chooses two keywords kw_0, kw_1 and sends kw_0, kw_1 to the challenger. The challenger first picks $b \xleftarrow{\$} \{0,1\}$, and then picks a word $W_1 \in \mathcal{L}$ randomly with the witness w_1 and generates the PEKS ciphertext CT^*_{kw} of kw_b as follows.

$$x_1 = \mathsf{ProjHash}(P, pk_{FS}, W_1, w_1), y_1 = \mathsf{ProjHash}(P, pk_{BS}, W_1, w_1),$$

$$C_1 = x_1 \circledast y_1 \circledast \Gamma(kw_b).$$

The challenger sets $CT^*_{kw} =< W_1, C_1 >$ as the keyword ciphertext and sends CT^*_{kw} to $\mathcal{A}$.

– Test Query-II. The procedure is the same as that in Test Query-I.

– Output. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ on b and wins the game if $b = b'$.

We define the advantage of $\mathcal{A}$ in Game0 as $\mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{FS},\mathcal{A}}(\lambda)$ and have that

$$\mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{FS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{SS-CKA}}_{\mathcal{FS},\mathcal{A}}(\lambda)$$

as Game0 strictly follows the SS-CKA model.

Game1. Let Game1 be the same game as Game0, except that the challenger chooses $y_1 \xleftarrow{\$} \mathcal{Y}$ instead of computing y_1 as $\mathsf{ProjHash}(P, pk_{BS}, W_1, w_1)$. Due to the correctness and pseudo-randomness of $\mathcal{SPHF}$, that is, the distribution $\{(P, W_1, pk_{BS}, y_1) | y_1 = \mathsf{ProjHash}(P, pk_{BS}, W_1, w_1)\}$ is computationally indistinguishable from the distribution $\{(P, W_1, pk_{BS}, y_1) | y_1 \xleftarrow{\$} \mathcal{Y}\}$, we have that

$$|\mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{FS},\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{FS},\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda).$$

Game2. Let Game2 be the same game as Game1, except that the challenger chooses $C_1 \xleftarrow{\$} \mathcal{Y}$ instead of computing $C_1 = x_1 \circledast y_1 \circledast \Gamma(kw_b)$. We can see that

$$\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{FS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{FS},\mathcal{A}}(\lambda).$$

It is easy to see that the adversary in Game2 can only win with probability 1/2 as C_1 is independent of b. Therefore, we have that $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{DS}\text{-}\mathcal{PEKS},\mathcal{A}}(\lambda) = 0$.

Therefore, from Game0, Game1 and Game2, we have that

$$|\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{FS},\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{SS-CKA}}_{\mathcal{FS},\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda).$$

As $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{FS},\mathcal{A}}(\lambda) = 0$ and $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda)$ is negligible, we have that $\mathsf{Adv}^{\mathsf{SS-CKA}}_{\mathcal{FS},\mathcal{A}}(\lambda)$ is also negligible, which completes the proof.

Lemma 2. *For any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SS\text{-}CKA}}_{\mathcal{BS},\mathcal{A}}(\lambda)$ is a negligible function.*

The proof of **Lemma 2.** can be easily obtained by following the proof of **Lemma 1.**, and hence is omitted.

Theorem 2. *The generic construction $\mathcal{DS}$-$\mathcal{PEKS}$ is secure against keyword guessing attack.*

The above theorem can be obtained from the following lemmas.

Lemma 3. *For any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{IND\text{-}KGA}}_{\mathcal{FS},\mathcal{A}}(\lambda)$ is a negligible function.*

Lemma 4. *For any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{IND\text{-}KGA}}_{\mathcal{BS},\mathcal{A}}(\lambda)$ is a negligible function.*

The proofs of **Lemma 3.** and **Lemma 4.** are similar to those of **Lemma 1.** and **Lemma 2.** as the generation of a trapdoor is the same as that of a PEKS ciphertext, and the security model of IND-KGA is also similar to that of SS-CKA. Therefore, we omit the proof details here. For the security against the keyword guessing attack-II, we have the following lemma.

Lemma 5. *For any polynomial-time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}}(\lambda)$ is a negligible function.*

Proof. In the IND-KGA-II game, if $b_1 = b_2$, then it is easy to see that the adversary has no advantage since the two keywords are canceled out in the internal testing-state C_{ITS}, which means C_{ITS} is independent of the keywords. In the following, we focus on the case that $b_1 \neq b_2$.

Here, we use the game-hopping technique again to prove this lemma. We define a sequence of attack games as follows.

$\mathsf{Game0}$. Let the original IND-CKA game be $\mathsf{Game0}$.

- Setup. The challenger runs the $\mathsf{Setup}, \mathsf{KeyGen}$ to generate system parameter P, key pairs (pk_{FS}, sk_{FS}) and (pk_{BS}, sk_{BS}). It gives adversary $\mathcal{A}$ the key pairs $(P, pk_{FS}, pk_{BS}, sk_{BS})$.
- $\mathsf{Challenge}$. $\mathcal{A}$ chooses challenge keywords kw_0, kw_1, kw_2 adaptively and sends them to the challenger. The challenger firstly picks $\{b_1, b_2\} \subset \{0, 1, 2\}$ randomly.
  The challenger picks two words $W_1, W_2 \in \mathcal{L}$ randomly with the witness w_1, w_2 respectively and generates the internal testing-state C_{ITS} as follows.

$$x_1 = \mathsf{ProjHash}(P, pk_{FS}, W_1, w_1), y_1 = \mathsf{ProjHash}(P, pk_{BS}, W_1, w_1),$$

$$x_2 = \mathsf{ProjHash}(P, pk_{FS}, W_2, w_2), y_2 = \mathsf{ProjHash}(P, pk_{BS}, W_2, w_2),$$

$$W = W_1 \circledcirc W_2, x = \mathsf{Hash}(P, sk_{FS}, W),$$

$$C = x_1 \circledast x_2 \circledast y_1 \circledast y_2 \circledast x^{-1} \circledast \Gamma(kw_{b_1}) \circledast \Gamma(kw_{b_2})^{-1},$$

$$W^* = \Delta w \otimes W, C^* = \Delta w \bullet C.$$

  Set $C^*_{ITS} =< W^*, C^* >$ and return C^*_{ITS} to $\mathcal{A}$.

– Output. Finally, $\mathcal{A}$ outputs its guess on $\{b_1, b_2\}$ as $\{b'_1, b'_2\} \subset \{0, 1, 2\}$ and wins the game if $\{b'_1, b'_2\} = \{b_1, b_2\}$.

We define the advantage of $\mathcal{A}$ in $\mathsf{Game0}$ as $\mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{BS},\mathcal{A}}(\lambda)$ and have that

$$\mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}}(\lambda)$$

as $\mathsf{Game0}$ strictly follows the IND-KGA-II model.

$\mathsf{Game1}$. Let $\mathsf{Game1}$ be the same game as $\mathsf{Game0}$, except that the challenger chooses $y \xleftarrow{\$} \mathcal{Y}$ and computes C^* as follows.

$$C^* = (x_1 \circledast x_2 \circledast y_1 \circledast y_2 \circledast x^{-1}) \circledast y.$$

In other words, the challenger replaces the part $\Delta w \cdot (\Gamma(kw_{b_1}) \circledast \Gamma(kw_{b_2})^{-1})$ with a random chosen element $y \in \mathcal{Y}$ during the generation of C^*. We now prove that the replacement in this way can make at most a negligible difference, that is,

Claim. For any polynomial-time adversary $\mathcal{A}$,

$$|\mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{BS},\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{BS},\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda).$$

Proof. Since the language $\mathcal{L}$ is a linear and homomorphic language, we have that the witness of W^* is $w^* = \Delta w \otimes w$ where w is the witness of W. Then based on our definition of LH-SPHF there exists a projection key hp' that

$$\mathsf{ProjHash}(P, \mathsf{hp}', W, w) = \Gamma(kw_{b_1}) \circledast \Gamma(kw_{b_2})^{-1}.$$

As $\mathcal{SPHF}$ is a Lin-Hom SPHF, we have that

$$\begin{aligned}\mathsf{ProjHash}(P, \mathsf{hp}', W^*, w^*) &= \Delta w \bullet \mathsf{ProjHash}(P, \mathsf{hp}', W, w)\\ &= \Delta w \bullet (\Gamma(kw_{b_1}) \circledast \Gamma(kw_{b_2})^{-1}).\end{aligned}$$

Moreover, the distribution $\{(P, W^*, \mathsf{hp}', y) | y = \mathsf{ProjHash}(P, \mathsf{hp}', W^*, w^*)\}$ is computationally indistinguishable from the distribution $\{(P, W^*, \mathsf{hp}', y) | y \xleftarrow{\$} \mathcal{Y}\}$ due to the correctness and pseudo-randomness of $\mathcal{SPHF}$. Therefore, we have that

$$|\mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{BS},\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Game0}}_{\mathcal{BS},\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda).$$

$\mathsf{Game2}$. Let $\mathsf{Game2}$ be the same game as $\mathsf{Game1}$, except that the challenger chooses $C^* \xleftarrow{\$} \mathcal{Y}$. We can see that

$$\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{BS},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Game1}}_{\mathcal{BS},\mathcal{A}}(\lambda).$$

It is easy to see that the adversary can only win in the $\mathsf{Game2}$ with probability $1/3$ as C^* is independent of b_1, b_2. Therefore, we have that $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{BS},\mathcal{A}}(\lambda) = 0$.

Therefore, from $\mathsf{Game0}, \mathsf{Game1}$ and $\mathsf{Game2}$, we have that

$$|\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{BS},\mathcal{A}}(\lambda) - \mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}}(\lambda)| \le \mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda).$$

As $\mathsf{Adv}^{\mathsf{Game2}}_{\mathcal{BS},\mathcal{A}}(\lambda) = 0$ and $\mathsf{Adv}^{\mathsf{PR}}_{\mathcal{SPHF},\mathcal{A}}(\lambda)$ is negligible, we have that $\mathsf{Adv}^{\mathsf{IND\text{-}KGA\text{-}II}}_{\mathcal{BS},\mathcal{A}}(\lambda)$ is also a negligible function, which proves the lemma.

5 Conclusion

In this paper, we proposed a new framework, named Dual-Server Public Key Encryption with Keyword Search (DS-PEKS), that can prevent the inside keyword guessing attack which is an inherent vulnerability of the traditional PEKS framework. We then introduced a new Smooth Projective Hash Function (SPHF) and used it to construct a generic DS-PEKS scheme. The new variant of SPHF introduced in this paper is of independent interest.

References

1. Abdalla, M., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 214–234. Springer, Heidelberg (2013)
3. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009)
4. Baek, J., Safavi-Naini, R., Susilo, W.: On the integration of public key data encryption and public key encryption with keyword search. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 217–232. Springer, Heidelberg (2006)
5. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part I. LNCS, vol. 5072, pp. 1249–1259. Springer, Heidelberg (2008)
6. Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient UC-secure authenticated key-exchange for algebraic languages. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 272–291. Springer, Heidelberg (2013)
7. Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New smooth projective hash functions and one-round authenticated key exchange. IACR Cryptology ePrint Archive **2013**, 34 (2013)
8. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013)
9. Blazy, O., Pointcheval, D., Vergnaud, D.: Round-optimal privacy-preserving protocols with smooth projective hash functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 94–111. Springer, Heidelberg (2012)
10. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
11. 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: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 75–83. Springer, Heidelberg (2006)

12. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
13. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
14. Di Crescenzo, G., Saraswat, V.: Public key encryption with searchable keywords based on jacobi symbols. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 282–296. Springer, Heidelberg (2007)
15. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, Eli (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
16. Halevi, S., Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. J. Cryptology **25**(1), 158–193 (2012)
17. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Computer Communications **32**(2), 394–396 (2009)
18. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011)
19. Khader, D.: Public key encryption with keyword search based on K-resilient IBE. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3982, pp. 298–308. Springer, Heidelberg (2006)
20. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. Journal of Systems and Software **83**(5), 763–771 (2010)
21. Rhee, H.S., Susilo, W., Kim, H.: Secure searchable public key encryption scheme against keyword guessing attacks. IEICE Electronic Express **6**(5), 237–243 (2009)
22. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA (2004)
23. Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 100–105. Springer, Heidelberg (2008)