Dual-Server Public-Key Encryption with Keyword Search for Secure Cloud Storage

Rongmao Chen*, Yi Mu*, *Senior Member, IEEE,* Guomin Yang, *Member, IEEE,* Fuchun Guo and Xiaofen Wang

Abstract—Searchable encryption is of increasing interest for protecting the data privacy in secure searchable cloud storage. In this work, we investigate the security of a well-known cryptographic primitive, namely Public Key Encryption with Keyword Search (PEKS) which is very useful in many applications of cloud storage. Unfortunately, it has been shown that the traditional PEKS framework suffers from an inherent insecurity called inside Keyword Guessing Attack (KGA) launched by the malicious server. To address this security vulnerability, we propose a new PEKS framework named Dual-Server Public Key Encryption with Keyword Search (DS-PEKS). As another main contribution, we define a new variant of the Smooth Projective Hash Functions (SPHFs) referred to as linear and homomorphic SPHF (LH-SPHF). We then show a generic construction of secure DS-PEKS from LH-SPHF. To illustrate the feasibility of our new framework, we provide an efficient instantiation of the general framework from a DDH-based LH-SPHF and show that it can achieve the strong security against inside KGA.

Index Terms—Keyword search, secure cloud storage, encryption, inside keyword guessing attack, smooth projective hash function, Diffie-Hellman language.

1 Introduction

Cloud storage outsourcing has become a popular application for enterprises and organizations to reduce the burden of maintaining big data in recent years. However, in reality, end users may not entirely trust the cloud storage servers and may prefer to encrypt their data before uploading them to the cloud server in order to protect the data privacy. This usually makes the data utilization more difficult than the traditional storage where data is kept in the absence of encryption. One of the typical solutions is the searchable encryption which allows the user to retrieve the encrypted documents that contain the user-specified keywords, where given the keyword trapdoor, the server can find the data required by the user without decryption.

Searchable encryption can be realized in either symmetric or asymmetric encryption setting. In [2], Song et al. proposed keyword search on ciphertext, known as Searchable Symmetric Encryption (SSE) and afterwards several SSE schemes [3], [4] were designed for improvements. Although SSE schemes enjoy high efficiency, they suffer from complicated secret key distribution. Precisely, users have to securely share secret keys which are used for data encryption. Otherwise they are not able to share the encrypted data outsourced to the cloud. To resolve this problem, Boneh et al. [5] introduced a more flexible primitive, namely Public Key Encryption with Keyword Search (PEKS) that enables a user to search encrypted data in the asymmetric encryption setting. In a PEKS system, using the receiver's public key, the 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.

1.1 Motivation of This Work

Despite of being free from secret key distribution, PEKS schemes suffer from an inherent insecurity regarding the trapdoor keyword privacy, namely *inside Keyword Guessing Attack* (KGA). The reason leading to such a security vulnerability is that anyone who knows receiver's public key can generate the PEKS ciphertext of arbitrary keyword himself. 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 schemes since the keyword space is roughly the same as a normal dictionary (e.g., all the meaningful English words), which has a much

- *R. Chen, Y. Mu, G. Yang , F. Guo and X. Wang are with the Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Australia. R. Chen is also with the College of Computer, National University of Defense Technology, China. X. Wang is also with School of Computer Science and Engineering and Big Data Research Center, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, China. Email: rc517@uowmail.edu.au, {ymu,gyang,fuchun}@uow.edu.au, wangxuedou@sina.com*
- **Corresponding authors. Tel: +61 2 4221 5228; Fax: +61 2 4221 4170.*

A preliminary version [1] of this paper was presented at the 20th Australasian Conference on Information Security and Privacy (ACISP'15).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2015.2510822, IEEE Transactions on Information Forensics and Security

smaller size than a password dictionary (e.g., all the words containing 6 alphanumeric characters). It is worth noting that in SSE schemes, only secret key holders can generate the keyword ciphertext and hence the adversarial server is not able to launch the inside KGA. As the keyword always indicates the privacy of the user data, it is therefore of practical importance to overcome this security threat for secure searchable encrypted data outsourcing.

1.2 Our Contributions

The contributions of this paper are four-fold.

- We formalize a new PEKS framework named *Dual-Server Public Key Encryption with Keyword Search* (DS-PEKS) to address the security vulnerability of PEKS.
- A new variant of *Smooth Projective Hash Function* (SPHF), referred to as *linear and homomorphic SPHF*, is introduced for a generic construction of DS-PEKS.
- We show a generic construction of DS-PEKS using the proposed Lin-Hom SPHF.
- To illustrate the feasibility of our new framework, an efficient instantiation of our SPHF based on the Diffie-Hellman language is presented in this paper.

1.3 Related Work

In this subsection, we describe a classification of PEKS schemes based on their security.

Traditional PEKS. Following Boneh et al.'s seminal work [5], Abdalla et al. [8] 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 [7] 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 [9] 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 Di Crescenzo and Saraswat [11]. The construction is derived from Cock's IBE scheme [12] which is not very practical.

Secure Channel Free PEKS. The original PEKS scheme [5] requires a secure channel to transmit the trapdoors. To overcome this limitation, Baek et al. [13] proposed a new PEKS scheme without requiring a secure channel, which is referred to as a secure channel-free PEKS (SCF-PEKS). The idea is to add the server's public/private key pair into a PEKS system. The keyword ciphertext and trapdoor are generated using the server's public key and hence only the server (designated tester) is able to perform the search. Rhee *et al.* [14] later enhanced Baek *et al.*'s security model [13] for SCF-PEKS where the attacker is allowed to obtain the relationship between the non-challenge ciphertexts and the trapdoor. They also presented an SCF-PEKS scheme secure under the enhanced security model in the random oracle model. Another extension on SCF-PEKS is by Emura et al. [15]. They enhanced the security model by introducing the *adaptively secure* SCF-PEKS, wherein an adversary is allowed to issue test queries adaptively.

Against Outside KGA. Byun et al. [16] 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*. [5] was susceptible to keyword guessing attack. Inspired by the work of Byun *et al.* [16], Yau *et al*. [17] 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 [13], [18]. The first PEKS scheme secure against outside keyword guessing attacks was proposed by Rhee et al. [19]. 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. Fang et al. [21] proposed a concrete SCF-PEKS scheme with (outside) KGA resilience. Similar to the work in [15], they also considered the adaptive test oracle in their proposed security definition.

Against Inside KGA. Nevertheless, all the schemes mentioned above are found to be vulnerable to keyword guessing attacks from a malicious server (i.e., inside KGA). Jeong et al. [22] showed a negative result that the consistency/correctness of PEKS implies insecurity to inside KGA in PEKS. Their result indicates that constructing secure and consistent PEKS schemes against inside KGA is impossible under the original framework. A potential solution is to propose a new framework of PEKS. In [10], Peng et al. proposed the notion of Public-key Encryption with Fuzzy Keyword Search (PEFKS) where each keyword corresponds to an exact trapdoor and a fuzzy trapdoor. The server is only provided with the fuzzy trapdoor and thus can no longer learn the exact keyword since two or more keywords share the same fuzzy keyword trapdoor. However, their scheme suffers from several limitations regarding the security and efficiency. On one hand, although the server cannot exactly guess the keyword, it is still able to know which small set the underlying keyword belongs to and thus the keyword privacy is not well preserved from the server. On the other hand, their scheme is impractical as the receiver has to locally find the matching ciphertext by using the exact trapdoor to filter out the non-matching ones from the set returned from the server.

Differences Between This Work and Its Preliminary Version [1]. Portions of the work presented in this paper have previously appeared as an extended abstract [1]. Compared to [1], we have revised and enriched the work substantially in the following aspects. First, in the preliminary work [1] where our generic DS-PEKS construction was presented, we showed neither a concrete construction of the linear and homomorphic SPHF nor a practical instantiation of the DS-PEKS framework. To fill this gap and illustrate the feasibility of the framework, in this paper (Section 6), we first show that a linear and homomorphic language $\mathcal{L}_{DH}$ can be derived from the Diffie-Hellman assumption and then construct a concrete linear and homomorphic SPHF, referred to as $\mathcal{SPHF}_{DH}$, from $\mathcal{L}_{DH}$. We provide a formal proof that $\mathcal{SPHF}_{DH}$ is *correct*, *smooth* and *pseudo-random*

and hence can be used for the instantiation of our generic construction. We then present a concrete DS-PEKS scheme from $\mathcal{SPHF}_{DH}$. To analyze its performance, we first give a comparison between existing schemes and our scheme and then evaluate its performance in experiments. We also revised the preliminary version [1] to enhance the presentation and readability. In the related work part, compared to the preliminary version, we add more literatures and give a clearer classification of the existing schemes based on their security. We present the security models of DS-PESK as experiments to make them more readable. Moreover, to make the concepts of SPHF and our newly defined variant clearer, we add Fig. 4 and Fig. 5 to highlight their key properties.

1.4 Organization

In Section 2, We propose a 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)in In Section 3. A generic construction of DS-PEKS from LH-SPHF is shown in Section 5 with formal correctness analysis and security proofs. Finally, we present an efficient instantiation of DS-PEKS from SPHF based on a language defined by the Diffie-Hellman problem in Section 6. We also analyze the performance of our scheme through comparisons with existing works and experimental evaluation.

2 A New Framework for PEKS

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

2.1 Definition of DS-PEKS

A DS-PEKS scheme mainly 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], [13], 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 a 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], [13]. 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 attack.

In the DS-PEKS system, 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. The formal definition of DS-PEKS is as follows.

Definition 1 (DS-PEKS). *A DS-PEKS scheme is defined by the following algorithms.*

- $\mathsf{Setup}(1^\lambda)$. *Takes as input the security parameter λ, generates the system parameters P;*
- $\mathsf{KeyGen}(P)$. *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;*
- $\mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_1)$. *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;*
- $\mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw_2)$. *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};*
- $\mathsf{FrontTest}(P, sk_{FS}, CT_{kw_1}, T_{kw_2})$. *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};*
- $\mathsf{BackTest}(P, sk_{BS}, C_{ITS})$. *Takes as input P, the back server's secret key sk_{BS} and the internal testing-state C_{ITS}, outputs the testing result 0 or 1;*

Correctness. It is required that for any keyword kw_1, kw_2, and $CT_{kw_1} \leftarrow \mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_1)$, $T_{kw_2} \leftarrow \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw_2)$, we have

$$\mathsf{BackTest}(P, sk_{BS}, C_{ITS}) = \begin{cases} 1 & kw_1 = kw_2, \\ 0 & kw_1 \neq kw_2. \end{cases}$$

where $C_{ITS} \leftarrow \mathsf{FrontTest}(P, sk_{FS}, CT_{kw_1}, T_{kw_2})$.

2.2 Security Models

In this subsection, we formalise the following security models for a DS-PEKS scheme against the adversarial front and back servers, respectively.

One should note that both the front server and the back server here are supposed to be "honest but curious" and will not collude with each other. More precisely, both the servers perform the testing strictly following the scheme procedures but may be curious about the underlying keyword. 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. We introduce two games, namely semantic-security against chosen keyword attack and indistinguishability against keyword guessing attack[1]

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2015.2510822, IEEE Transactions on Information Forensics and Security

Experiment $\mathsf{EXP}^{\mathsf{SS\text{-}CKA}}_{DS\text{-}PEKS,\mathcal{A}}(\lambda)$
$KWSet \leftarrow \varnothing$;
$(pk_{FS}, sk_{FS}, pk_{BS}, sk_{BS}) \xleftarrow{\$} \mathsf{KeyGen}(P)$;
$\{kw_0, kw_1, \mathsf{state}\} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_T(\cdot)}(\mathsf{F}, pk_{FS}, sk_{FS}, pk_{BS})$;
$b \xleftarrow{\$} \{0,1\}$;
$CT^*_{kw} \xleftarrow{\$} \mathsf{DS\text{-}PEKS}(pk_{FS}, pk_{BS}, kw_b)$;
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_T(\cdot)}(\mathsf{G}, CT^*_{kw}, \mathsf{state})$;
If $kw_0 \notin KWSet \wedge kw_1 \notin KWSet$, then return b';
Otherwise return 0.

Oracle $\mathcal{O}_T(CT_{kw}, kw)$
$KWSet \leftarrow KWSet \cup \{kw\}$;
$T_{kw} \xleftarrow{\$} \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw)$;
$C_{ITS} \xleftarrow{\$} \mathsf{FrontTest}(P, sk_{FS}, CT_{kw}, T_{kw})$;
Return $0/1 \leftarrow \mathsf{BackTest}(P, sk_{BS}, C_{ITS})$.

Fig. 1. SS-CKA Experiment for Adversarial Front Server

Experiment $\mathsf{EXP}^{\mathsf{IND\text{-}KGA}}_{DS\text{-}PEKS,\mathcal{A}}(\lambda)$
$KWSet \leftarrow \varnothing$;
$(pk_{FS}, sk_{FS}, pk_{BS}, sk_{BS}) \xleftarrow{\$} \mathsf{KeyGen}(P)$;
$\{kw_0, kw_1, \mathsf{state}\} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_T(\cdot)}(\mathsf{F}, pk_{FS}, sk_{FS}, pk_{BS})$;
$b \xleftarrow{\$} \{0,1\}$;
$T^*_{kw} \xleftarrow{\$} \mathsf{DS\text{-}Trapdoor}(pk_{FS}, pk_{BS}, kw_b)$;
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_T(\cdot)}(\mathsf{G}, T^*_{kw}, \mathsf{state})$;
If $kw_0 \notin KWSet \wedge kw_1 \notin KWSet$, then return b';
Otherwise return 0.

Oracle $\mathcal{O}_T(CT_{kw}, kw)$
$KWSet \leftarrow KWSet \cup \{kw\}$;
$T_{kw} \xleftarrow{\$} \mathsf{DS\text{-}Trapdoor}(P, pk_{FS}, pk_{BS}, kw)$;
$C_{ITS} \xleftarrow{\$} \mathsf{FrontTest}(P, sk_{FS}, CT_{kw}, T_{kw})$;
Return $0/1 \leftarrow \mathsf{BackTest}(P, sk_{BS}, C_{ITS})$.

Fig. 2. IND-KGA Experiment for Adversarial Front Server

to capture the security of PEKS ciphertext and trapdoor, respectively.

I. *Semantic-Security against Chosen Keyword Attack*. 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.

Formally, we introduce an experiment in Fig. 1 for the SS-CKA security definition against the adversarial front server. In the experiment, the adversary $\mathcal{A}$ is given the public/private key pair of the front server and the public key of the back server. In the find phase, $\mathcal{A}$ can test any pair of PEKS ciphertext and keyword by querying the oracle $\mathcal{O}_T$ and eventually output two challenging keywords (kw_0, kw_1) with the hint information "state". With a random bit $b \in \{0,1\}$ as input, the experiment generates and then sends the PEKS ciphertext CT^*_{kw} of keyword kw_b to $\mathcal{A}$. During the guess phase, $\mathcal{A}$ can continue the query to $\mathcal{O}_T$ and finally output its guess b'. The guess b' is a valid output of the experiment if and only if that $\mathcal{A}$ has never queried $\mathcal{O}_T$ with the challenge keywords. We refer to such an adversarial front server $\mathcal{A}$ in the above experiment as an SS-CKA adversary and define its advantage as

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

II. *Indistinguishability against Keyword Guessing Attack*. This security model captures that the trapdoor reveals no information about the underlying keyword to the adversarial front server. We define the security experiment as shown in Fig. 2. The experiment is similar to that of SS-CKA experiment except that in the challenge phase, the adversary is given the trapdoor instead of the PEKS ciphertext.

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

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

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.

III. *Semantic-Security against Chosen Keyword Attack.* Here the SS-CKA experiment 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 experiment as an SS-CKA adversary and define its advantage as

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

IV. *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 experiment as an IND-KGA adversary and define its advantage as

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

V. *Indistinguishability against Keyword Guessing Attack-II.* In our defined security notion of IND-KGA-II, as shown in Fig. 3, it is required that a malicious back server cannot learn any information about the underlying two keywords involved in the internal testing-state. First of all, we should note that both keywords involved in the internal-testing state plays the same role regardless of their initial source (i.e., from the PEKS ciphertext or the trapdoor). Therefore, the task of the adversary is to guess the two underlying keywords in the internal testing state as a whole, instead of each one in the initial PEKS ciphertext and the initial trapdoor. Therefore, it is insufficient for the adversary to submit only two challenge keywords and hence we require the adversary to submit three different keywords in the challenge stage and guess which two keywords are chosen given the challenge internal-testing state.

Formally, in the experiment, the adversary $\mathcal{A}$ is given the public key of the front server and the public/private key pair of the back server. In the challenge phase, the adversary outputs three challenging keywords (kw_0, kw_1, kw_2). With

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2015.2510822, IEEE Transactions on Information Forensics and Security

Experiment $\mathsf{EXP}^{\mathsf{IND\text{-}KGA\text{-}II}}_{DS\text{-}PEKS,\mathcal{A}}(\lambda)$
$(pk_{FS}, sk_{FS}, pk_{BS}, sk_{BS}) \xleftarrow{\$} \mathsf{KeyGen}(P)$;
$\{kw_0, kw_1, kw_2, \mathsf{state}\} \xleftarrow{\$} \mathcal{A}(pk_{FS}, pk_{BS}, sk_{BS})$;
$b_1 \xleftarrow{\$} \{0,1,2\}, b_2 \xleftarrow{\$} \{0,1,2\}$;
$CT^*_{kw} \xleftarrow{\$} \mathsf{DS\text{-}PEKS}(pk_{FS}, pk_{BS}, kw_{b_1})$;
$T^*_{kw} \xleftarrow{\$} \mathsf{DS\text{-}Trapdoor}(pk_{FS}, pk_{BS}, kw_{b_2})$;
$C^*_{ITS} \leftarrow \mathsf{FrontTest}(sk_{FS}, CT^*_{kw}, T^*_{kw})$;
$\{b'_1, b'_2\} \xleftarrow{\$} \mathcal{A}(\mathsf{G}, C^*_{ITS}, \mathsf{state})$;
Return $\{b'_1, b'_2\}$;

Fig. 3. IND-KGA-II Experiment for Adversarial Back Server

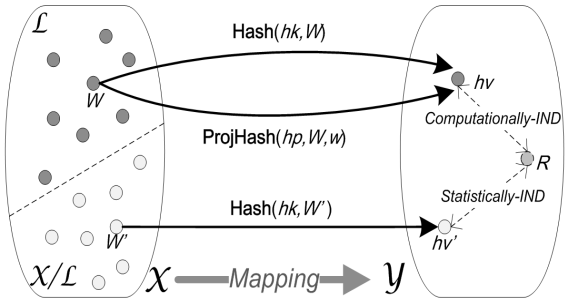


Fig. 4. Smooth Projective Hash Function

two random bits $b_1 \in \{0,1,2\}, b_2 \in \{0,1,2\}$ as input, the experiment first generates the PEKS ciphertext CT^*_{kw} of kw_{b_1} and the trapdoor T^*_{kw} of kw_{b_2}, it finally generates and sends the internal-testing state C^*_{ITS} to $\mathcal{A}$. In the guess phase, $\mathcal{A}$ finally output its guess $\{b'_1, b'_2\}$. We refer to such an adversary $\mathcal{A}$ in the above experiment as an IND-KGA-II adversary and define its advantage as,

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

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

Definition 2 (Security of DS-PEKS). *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 [23]. We start with the original definition of an SPHF.

3.1 Original Definition of SPHFs

As illustrated in Fig. 4, an SPHF is 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., $\mathcal{L} \subset \mathcal{X}$. Formally, an SPHF system over a language $\mathcal{L} \subset \mathcal{X}$, onto a set $\mathcal{Y}$, is defined by the following five algorithms ($\mathsf{SPHFSetup}$, HashKG, ProjKG, 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 $W \in \mathcal{L}$.

The *correctness* of an SPHF requires that for a word $W \in \mathcal{L}$ with w the witness,

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

Another property of SPHFs is *smoothness*, which means that for any $W \in \mathcal{X} \backslash \mathcal{L}$, the following two distributions are statistically indistinguishable :

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

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

In summary, an SPHF has the property that the projection key uniquely determines the hash value of any word in the language $\mathcal{L}$ but gives almost no information about the hash value for any point in $\mathcal{X} \setminus \mathcal{L}$.

In this paper, we require another important property of smooth projective hash functions that was introduced in [6]. Precisely, we require the SPHF to be *pseudo-random*. That is, 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.

3.2 A New Variant–Linear and Homomorphic SPHFs

In this paper, we introduce a new variant of smooth projective hash function. In addition to the original properties, we consider two new properties – *linear* and *homomorphic*, which are defined below.

Let $\mathcal{W}$ be the witness space of $\mathcal{L}$. We first describe the operations on the sets $< \mathcal{L}, \mathcal{Y}, \mathcal{W} >$ as follows.

1) $\circledcirc : \mathcal{L} \times \mathcal{L} \to \mathcal{L}$. $\forall\, W_1, W_2 \in \mathcal{L}, W_1 \circledcirc W_2 \in \mathcal{L}$;
2) $\circledast : \mathcal{Y} \times \mathcal{Y} \to \mathcal{Y}$. $\forall\, y_1, y_2 \in \mathcal{Y}, y_1 \circledast y_2 \in \mathcal{Y}$;
3) $\odot, \oplus : \mathcal{W} \times \mathcal{W} \to \mathcal{W}$. $\forall w_1, w_2 \in \mathcal{W}$, $w_1 \odot w_2 \in \mathcal{W}$ and $w_1 \oplus w_2 \in \mathcal{W}$;
4) $\otimes : \mathcal{W} \times \mathcal{L} \to \mathcal{L}$. $\forall\, w \in \mathcal{W}, \forall\, W \in \mathcal{L}, w \otimes W \in \mathcal{L}$;
5) $\bullet : \mathcal{W} \times \mathcal{Y} \to \mathcal{Y}$. $\forall\, w \in \mathcal{W}, \forall\, y \in \mathcal{Y}, w \bullet y \in \mathcal{Y}$.

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

As shown in Fig. 5, our new SPHF requires the underlying language to be also *linear and homomorphic language*, which is defined below.

Definition 3 (Linear and Homomorphic Language). *A language $\mathcal{L}$ is linear and homomorphic if it satisfies the following properties.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2015.2510822, IEEE Transactions on Information Forensics and Security

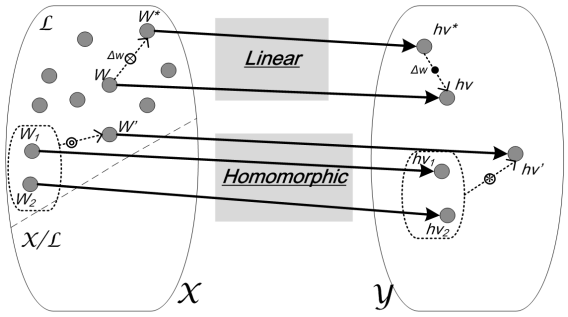


Fig. 5. Linear and Homomorphic SPHF

- *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 define the *Lin-Hom SPHF* as follows.

Definition 4 (Lin-Hom SPHF (LH-SPHF). *We say an SPHF is a Lin-Hom SPHF (LH-SPHF) if the underlying language $\mathcal{L}$ is a linear and homomorphic language and it 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*[2]

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

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

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

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*

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

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

$$\mathsf{ProjHash}(\mathsf{hp}, W', w') = \mathsf{ProjHash}(\mathsf{hp}, W_1, w_1) \circledast \mathsf{ProjHash}(\mathsf{hp}, W_2, w_2),$$

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 $\mathsf{ProjHash}(\mathsf{hp}, W, w) = y$. In Section 6.1, we will present a concrete LH-SPHF scheme based on the Diffie-Hellman problem, which has all these properties.

2. For simplicity, we omit the input of $(\mathcal{L}, \mathsf{param})$ for the algorithm in the rest of paper.

4 Generic Construction of DS-PEKS

4.1 Generic Construction

Let $\mathcal{SPHF} = (\mathsf{SPHFSetup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ be a 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 shown in Fig. 6.

Correctness Analysis. One can see that the correctness of this construction is guaranteed by the properties of the LH-SPHF. We give the analysis as follows.

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

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

Therefore,

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

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

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

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

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

In this subsection, we analyse the security of the above generic construction $\mathcal{DS}$-$\mathcal{PEKS}$. Due to the space limitation, we omit the proof details here and refer readers to [1] for a full security proof.

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

This conclusion is 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.*

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

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.*

$\mathsf{Setup}(1^\lambda)$. Take as input the security parameter λ, run the algorithm $\mathsf{SPHFSetup}$ 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.

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

$\mathsf{DS\text{-}PEKS}(P, pk_{FS}, pk_{BS}, kw_1)$. 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.

$$x_1 = \mathsf{ProjHash}(P, pk_{FS}, W_1, w_1), \quad y_1 = \mathsf{ProjHash}(P, pk_{BS}, W_1, w_1), \quad C_1 = x_1 \circledast y_1 \circledast \Gamma(kw_1).$$

Return $CT_{kw_1} =< W_1, C_1 >$ 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.

$$x_2 = \mathsf{ProjHash}(P, pk_{FS}, W_2, w_2), \quad y_2 = \mathsf{ProjHash}(P, pk_{BS}, W_2, w_2), \quad C_2 = x_2 \circledast y_2 \circledast \Gamma(kw_2)^{-1}.$$

Return $T_{kw_2} =< W_2, C_2 >$ 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.

$$W = W_1 \circledcirc W_2, \quad x = \mathsf{Hash}(P, sk_{FS}, W), \quad C = C_1 \circledast C_2 \circledast x^{-1}, \quad W^* = \Delta w \otimes W, C^* = \Delta w \bullet C.$$

Return $C_{ITS} =< W^*, C^* >$ 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.

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

If yes output 1, else output 0.

Fig. 6. A Generic Construction of DS-PEKS from Lin-Hom SPHF

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. 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.*

5 AN INSTANTIATION OF DS-PEKS

5.1 A Concrete LH-SPHF from DDH Assumption

We introduce the Diffie Hellman language $\mathcal{L}_{DH}$ and show how to derive an LH-SPHF. Let $\mathbb{G}$ be a group of primer order p and $g_1, g_2 \in \mathbb{G}$ the generators of $\mathbb{G}$.

Decision Diffie-Hellman (DDH) Assumption. The DDH assumption says that for any a, b, r randomly chosen from $\mathbb{Z}_p$, $\{(g_1, g_1^a, g_1^b, g_1^{ab})\}$ is computationally indistinguishable from $\{(g_1, g_1^a, g_1^b, g_1^r)\}$.

Diffie-Hellman Language. The Diffie-Hellman Language is defined as follows.

$$\mathcal{L}_{DH} = \{(u_1, u_2) | \exists r \in \mathbb{Z}_p, s.t., u_1 = g_1^r, u_2 = g_2^r\}$$

One can see that the witness space of $\mathcal{L}_{DH}$ is $\mathcal{W} = \mathbb{Z}_p$ and $\mathcal{L}_{DH} \subset \mathbb{G}^2$. We have the following theorems.

Theorem 3. The Diffie-Hellman language $\mathcal{L}_{DH}$ is a linear and homomorphic language.

Proof: We show that $\mathcal{L}_{DH}$ satisfies the properties of a linear and homomorphic language.

1). For a word $W = (g_1^r, g_2^r)$ with the witness $w = r \in \mathbb{Z}_p$ and $\Delta r \in \mathbb{Z}_p$, we have,

$$W^* = \Delta r \otimes W = (g_1^{r\Delta r}, g_2^{r\Delta r}) \in \mathcal{L}_{DH},$$

which has the witness $w^* = r\Delta r$.

2). For any two word $W_1 = (g_1^{r_1}, g_2^{r_1})$ (witness $w_1 = r_1$), $W_2 = (g_1^{r_2}, g_2^{r_2})$ (witness $w_2 = r_2$), we have,

$$W^* = W_1 \circledcirc W_2 = (g_1^{r_1+r_2}, g_2^{r_1+r_2}) \in \mathcal{L}_{DH},$$

which has the witness $w^* = w_1 \oplus w_2 = r_1 + r_2$. □

LH-SPHF on $\mathcal{L}_{DH}$. Here we show how to construct an LH-SPHF (denoted by $\mathcal{SPHF}_{DH}$) over the language $\mathcal{L}_{DH} \subset \mathcal{X} = \mathbb{G}^2$ onto the group $\mathcal{Y} = \mathbb{G}$. The concrete construction is as follows.

- $\mathsf{SPHFSetup}(1^\lambda)$: $\mathsf{param} = (\mathbb{G}, p, g_1, g_2)$;
- $\mathsf{HashKG}(\mathcal{L}_{DH}, \mathsf{param})$: $\mathsf{hk} = (\alpha_1, \alpha_2) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}))$: $\mathsf{hp} = g_1^{\alpha_1} g_2^{\alpha_2} \in \mathbb{Z}_p$;
- $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), W = (g_1^r, g_2^r))$: $\mathsf{hv} = g_1^{r\alpha_1} g_2^{r\alpha_2} \in \mathbb{Z}_p$;
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}_{DH}, \mathsf{param}), W = (g_1^r, g_2^r), w = r)$: $\mathsf{hv}' = \mathsf{hp}^r \in \mathbb{Z}_p$.

As for $\mathcal{SPHF}_{DH}$, we have the following theorems.

Theorem 4. $\mathcal{SPHF}_{DH}$ is a smooth projective hash function.

Proof: We show that $\mathcal{SPHF}_{DH}$ is of correctness, smoothness and pseudo-randomness.

1) *Correctness*. With the above notations, we have

$$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), W) = g_1^{r\alpha_1} g_2^{r\alpha_2} = \mathsf{hp}^r$$
$$= \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}_{DH}, \mathsf{param}), W, w).$$

2) *Smoothness*. Suppose $g_2 = g_1^{\theta}$. Note that $\mathsf{hp} = g_1^{\alpha_1} g_2^{\alpha_2}$ which constraints (α_1, α_2) to satisfy

$$\log_{g_1} \mathsf{hp} = \alpha_1 + \theta\alpha_2. \quad (1)$$

Let $W' = (g_1^{r_1}, g_2^{r_2}) \in \mathcal{X} \backslash \mathcal{L}_{DH}$ where $r_1 \neq r_2$, then the hash value hv_1 of W' is

$$\mathsf{hv}_1 = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), W') = g_1^{r_1\alpha_1} g_2^{r_2\alpha_2},$$

which also constraints (α_1, α_2) to satisfy

$$\log_{g_1} \mathsf{hv}_1 = r_1\alpha_1 + r_2\theta\alpha_2. \quad (2)$$

For the above two equations, we have

$$(\alpha_1, \alpha_2) \cdot \mathbf{A} = (\log_{g_1} \mathsf{hp}, \log_{g_1} \mathsf{hv}_1),$$

where $\mathbf{A}$ is a matrix defined as

$$\mathbf{A} = \begin{bmatrix} 1 & r_1 \\ \theta & r_2\theta \end{bmatrix}.$$

Since the determinant of $\mathbf{A}$ is $\theta \cdot (r_2 - r_1)$ that is nonzero ($r_1 \neq r_2$), we have that the equation (1) is independent of the equation (2). Therefore, we have that hv_1 is statistically indistinguishable from any element randomly chosen from $\mathbb{G}$.

3) *Pseudo-randomness*. One can easily obtain this property from the DDH assumption. We omit the proof details due to the space limitation.

□

Theorem 5. $\mathcal{SPHF}_{DH}$ is a Lin-Hom SPHF.

Proof: As shown previously, $\mathcal{L}_{DH}$ is a linear and homomorphic language. Now we show that $\mathcal{SPHF}_{DH}$ satisfies the following properties.

1) For a word $W = (g_1^r, g_2^r)$ with the witness $w = r \in \mathbb{Z}_p$ and $\Delta r \in \mathbb{Z}_p$, we have $\Delta r \otimes W = (g_1^{r\Delta r}, g_2^{r\Delta r})$. Therefore, we have

$$\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), \Delta r \otimes W) = g_1^{r\Delta r\alpha_1} g_2^{r\Delta r\alpha_2}$$
$$= (g_1^{r_1\alpha_1} g_2^{r_2\alpha_2})^{\Delta r} = \Delta r \bullet \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), W)$$

2) For any two word $W_1 = (g_1^{r_1}, g_2^{r_1})$, $W_2 = (g_1^{r_2}, g_2^{r_2})$, we have $W_1 \circledcirc W_2 = (g_1^{r_1+r_2}, g_2^{r_1+r_2}) \in \mathcal{L}_{DH}$. Therefore, we have

$$\begin{aligned} & \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}_{DH}, \mathsf{param}), W_1 \circledcirc W_2) \\ = \; & g_1^{(r_1+r_2)\cdot\alpha_1} \cdot g_2^{(r_1+r_2)\cdot\alpha_2} \\ = \; & \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_1) \\ & \circledast \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W_2). \end{aligned}$$

This proves the theorem. □

5.2 The Concrete Scheme

The concrete scheme based on $\mathcal{SPHF}_{DH}$ introduced above is as follows.

Setup. Let $\mathbb{G}$ be a group with prime order p and g_1, g_2 be two generators of $\mathbb{G}$. $H : \{0,1\}^* \to \mathbb{G}$ is a collision-resistant hash function. The system parameter is $(p, g_1, g_2, \mathbb{G}, H)$.

KeyGen. Pick $\alpha_1, \alpha_2, \beta_1, \beta_2$ from $\mathbb{Z}_p$ randomly and generate the public/secret key pair (pk_{FS}, sk_{FS}), (pk_{BS}, sk_{BS}) for the front server and the back server respectively as follows,

$$pk_{FS} = h_1 = g_1^{\alpha_1} g_2^{\alpha_2}, sk_{FS} = (\alpha_1, \alpha_2),$$
$$pk_{BS} = h_2 = g_1^{\beta_1} g_2^{\beta_2}, sk_{BS} = (\beta_1, \beta_2).$$

DS-PEKS. For a keyword kw_1, pick $r_1 \xleftarrow{\$} \mathbb{Z}_p$, generate the PEKS ciphertext of kw_1 as follows,

$$CT_{kw} = (g_1^{r_1},\ g_2^{r_1},\ h_1^{r_1} h_2^{r_1} H(kw_1)).$$

DS-Trapdoor. For a keyword kw_2, pick $r_2 \xleftarrow{\$} \mathbb{Z}_p$, generate the trapdoor of kw_2 as follows,

$$T_{kw} = (g_1^{r_2},\ g_2^{r_2},\ h_1^{r_2} h_2^{r_2} H(kw_2)^{-1}).$$

FrontTest. Pick $\gamma \xleftarrow{\$} \mathbb{Z}_p$ and compute the internal testing-state C_{ITS} as follows,

$$\begin{aligned} & CT_{kw} \cdot T_{kw'} = (C_1, C_2, C_3) \\ & = (g_1^{r_1+r_2},\ g_2^{r_1+r_2},\ h_1^{r_1+r_2} h_2^{r_1+r_2} (H(kw_1)H(kw_2)^{-1})), \\ & C_{ITS} = (C_1^*, C_2^*, C_3^*) = (C_1^{\gamma}, C_2^{\gamma}, (C_3/(C_1^{\alpha_1} C_2^{\alpha_2}))^{\gamma}). \end{aligned}$$

BackTest. For an internal testing-state $C_{ITS} = (C_1^*, C_2^*, C_3^*)$, do the testing as follows,

$$C_1^{*\beta_1} C_2^{*\beta_2} \stackrel{?}{=} C_3^*.$$

If the equation holds, outputs 1, otherwise outputs 0.

Correctness. It is easy to obtain the correctness as $C_1^{\alpha_1} C_2^{\alpha_2} = g_1^{(r_1+r_2)\alpha_1} g_2^{(r_1+r_2)\alpha_2} = h_1^{r_1+r_2}$ and we have that,

$$C_3^* = (C_3/(C_1^{\alpha_1} C_2^{\alpha_2}))^{\gamma} = h_2^{(r_1+r_2)\gamma} (H(kw_1)H(kw_2)^{-1})^{\gamma}.$$

Therefore, if $kw_1 = kw_2$, i.e.,$C_3^* = h_2^{(r_1+r_2)\gamma}$, then

$$\begin{aligned} C_1^{*\beta_1} C_2^{*\beta_2} &= C_1^{\gamma\beta_1} C_2^{\gamma\beta_2} = g_1^{\beta_1\cdot(r_1+r_2)\gamma} g_2^{\beta_2\cdot(r_1+r_2)\gamma} \\ &= h_2^{(r_1+r_2)\gamma} = C_3^*. \end{aligned}$$

Otherwise, the equation would hold due to the collision-resistance of H.

Security. The following corollary can be obtained directly from **Theorem 1-5**.

Corollary. The concrete construction is a secure DS-PEKS scheme.

5.3 Performance Evaluation

In this section, we first give a comparison between existing schemes and our scheme in terms of computation, size and security. We then evaluate its performance in experiments.

Computation Costs. As shown in Table 1, all the existing schemes [5], [10], [20] require the pairing computation during the generation of PEKS ciphertext and testing and hence are less efficient than our scheme, which does not need any pairing computation. In our scheme, the computation cost of PEKS generation, trapdoor generation and testing are $4\mathrm{Exp}_{\mathbb{G}_1}$+$1\mathrm{Hash}_{\mathbb{G}_1}$+$2\mathrm{Mul}_{\mathbb{G}_1}$, $4\mathrm{Exp}_{\mathbb{G}_1}$+$1\mathrm{Hash}_{\mathbb{G}_1}$+$2\mathrm{Mul}_{\mathbb{G}_1}$, and $7\mathrm{Exp}_{\mathbb{G}_1}$+$3\mathrm{Mul}_{\mathbb{G}_1}$ respectively, where $\mathrm{Exp}_{\mathbb{G}_1}$ denotes the computation of one exponentiation in $\mathbb{G}_1$, $\mathrm{Mul}_{\mathbb{G}_1}$ denotes the costs of one multiplication in $\mathbb{G}_1$, $\mathrm{Mul}_{\mathbb{G}_1}$ and $\mathrm{Hash}_{\mathbb{G}_1}$ respectively denote the cost of one multiplication and one hashing operation in $\mathbb{G}_1$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2015.2510822, IEEE Transactions on Information Forensics and Security

TABLE 1
Performance Comparisons between Existing Works and Our Scheme

Schemes	Computation			Size (bits)		Inside KGA
	PEKS Generation	*Trapdoor Generation*	*Testing*	*PEKS*	*Trapdoor*	
BCOP [5]	$2\text{Exp}_{\mathbb{G}_1}+2\text{Hash}_{\mathbb{G}_1}$ + $1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}$	$1\text{Hash}_{\mathbb{G}_1}+1\text{Exp}_{\mathbb{G}_1}$	$1\text{Hash}_{\mathbb{G}_1}+1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}$	$1\lvert\mathbb{G}_1\rvert+\log p$	$1\lvert\mathbb{G}_1\rvert$	$\times$
RPSL [20]	$2\text{Exp}_{\mathbb{G}_1}+2\text{Hash}_{\mathbb{G}_1}$ +1 $\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}$	$2\ \text{Hash}_{\mathbb{G}_1}+3\text{Exp}_{\mathbb{G}_1}$	$2\text{Hash}_{\mathbb{G}_1}+2\text{Exp}_{\mathbb{G}_1}$ +$1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}+1\text{Mul}_{\mathbb{G}_1}$	$1\lvert\mathbb{G}_1\rvert+\lambda$	$2\lvert\mathbb{G}_1\rvert$	$\times$
XJWW [10]	$4\text{Exp}_{\mathbb{G}_1}+4\text{Hash}_{\mathbb{G}_1}$ +2 $\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}$	$2\ \text{Hash}_{\mathbb{G}_1}+2\text{Exp}_{\mathbb{G}_1}$	$2\text{Hash}_{\mathbb{G}_1}+2\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_2}$	$3\lvert M\rvert+2\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{G}_1\rvert$	$\surd^*$(*weak*)
Our Scheme	$4\text{Exp}_{\mathbb{G}_1}+1\text{Hash}_{\mathbb{G}_1}$ +$2\text{Mul}_{\mathbb{G}_1}$	$4\text{Exp}_{\mathbb{G}_1}+1\text{Hash}_{\mathbb{G}_1}$ +$2\text{Mul}_{\mathbb{G}_1}$	$7\text{Exp}_{\mathbb{G}_1}+3\text{Mul}_{\mathbb{G}_1}$	$3\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{G}_1\rvert$	$\surd$

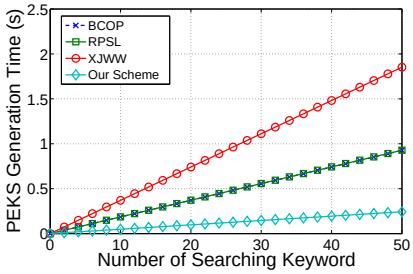


Fig. 7. Computation Cost of PEKS Generation in Different Schemes

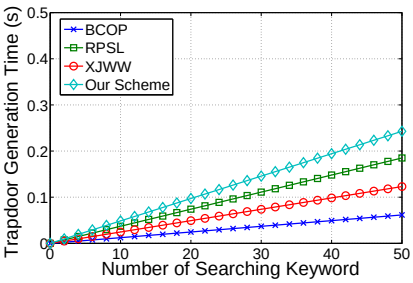


Fig. 8. Computation Cost of Trapdoor Generation in Different Schemes

Experiment Results. To evaluate the efficiency of schemes in experiments, we also implement the scheme utilizing the GNU Multiple Precision Arithmetic (GMP) library and Pairing Based Cryptography (PBC) library. The following experiments are based on coding language C on Linux system (more precise, 2.6.35-22-generic version) with an Intel(R) Core(TM) 2 Duo CPU of 3.33 GHZ and 2.00-GB RAM. For the elliptic curve, we choose an MNT curve with a base filed size of 159 bits and p=160 bits and $|q|$=80 bits.

As shown in Fig. 7, our scheme is the most efficient in terms of PEKS computation. It is because that our scheme does not include pairing computation. Particularly, the scheme [10] requires the most computation cost due to 2 pairing computation per PEKS generation. As for the trapdoor generation indicated in Figure 8, as all the existing schemes do not involve pairing computation, the computation cost is much lower than that of PEKS generation. It is worth noting that the trapdoor generation in our scheme is slightly higher than those of existing schemes due to the additional exponentiation computations. When the searching keyword number is 50, the total computation cost of our scheme is about 0.25 seconds. As illustrated in Fig. 9, the

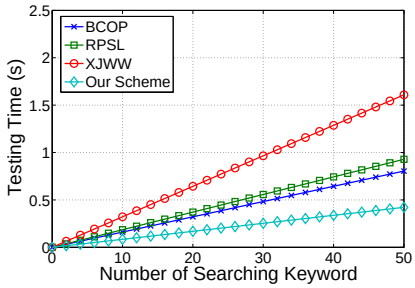


Fig. 9. Computation Cost of Testing in Different Schemes

scheme [10] cost the most time due to an additional pairing computation in the exact testing stage. One should note that this additional pairing computation is done on the user side instead of the server. Therefore, it could be the computation burden for users who may use a light device for searching data. In our scheme, although we also require another stage for the testing, our computation cost is actually lower than that of any existing scheme as we do not require any pairing computation and all the searching work is handled by the server.

6 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 also introduced a new Smooth Projective Hash Function (SPHF) and used it to construct a generic DS-PEKS scheme. An efficient instantiation of the new SPHF based on the Diffie-Hellman problem is also presented in the paper, which gives an efficient DS-PEKS scheme without pairings.

Acknowledgement

We thank the anonymous reviewers for their invaluable feedbacks on this work. The work of Guomin Yang is supported by the National Natural Science Foundation of China (Grant No. 61472308).

References

[1] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Information Security and Privacy - 20th Australasian Conference, ACISP*, 2015, pp. 59–76.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000, pp. 44–55.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-preserving encryption for numeric data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004, pp. 563–574.

[4] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, 2006, pp. 79–88.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT*, 2004, pp. 506–522.

[6] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," in *EUROCRYPT*, 2003, pp. 524–543.

[7] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *NDSS*, 2004.

[8] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *CRYPTO*, 2005, pp. 205–222.

[9] D. Khader, "Public key encryption with keyword search based on k-resilient IBE," in *Computational Science and Its Applications - ICCSA*, 2006, pp. 298–308.

[10] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[11] G. D. Crescenzo and V. Saraswat, "Public key encryption with searchable keywords based on jacobi symbols," in *INDOCRYPT*, 2007, pp. 282–296.

[12] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and Coding*, 2001, pp. 360–363.

[13] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications - ICCSA*, 2008, pp. 1249–1259.

[14] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *ASIACCS*, 2009, pp. 376–379.

[15] K. Emura, A. Miyaji, M. S. Rahman, and K. Omote, "Generic constructions of secure-channel free searchable encryption with adaptive security," *Security and Communication Networks*, vol. 8, no. 8, pp. 1547–1560, 2015.

[16] J. W. Byun, H. S. Rhee, H. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management, Third VLDB Workshop, SDM*, 2006, pp. 75–83.

[17] W. Yau, S. Heng, and B. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *ATC*, 2008, pp. 100–105.

[18] J. Baek, R. Safavi-Naini, and W. Susilo, "On the integration of public key data encryption and public key encryption with keyword search," in *Information Security ISC*, 2006, pp. 217–232.

[19] H. S. Rhee, W. Susilo, and H. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronic Express*, vol. 6, no. 5, pp. 237–243, 2009.

[20] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

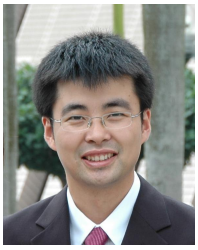
[21] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Inf. Sci.*, vol. 238, pp. 221–241, 2013.

[22] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Computer Communications*, vol. 32, no. 2, pp. 394–396, 2009.

[23] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *EUROCRYPT*, 2002, pp. 45–64.

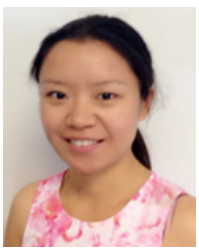
Rongmao Chen received the B.S. and M.S. degrees in computer science from National University of Defense Technology, China in 2011 and 2013 respectively, and is currently pursing the full-time PhD degree in the School of Computing and Information Technology, University of Wollongong, Australia. His major research interests include cryptography, data security and privacy in cloud computing, network security.

Yi Mu received his PhD from the Australian National University in 1994. He currently is professor and the co-director of Centre for Computer and Information Security Research at University of Wollongong, Australia. His current research interests include information security and cryptography. Yi Mu is the editor-in-chief of International Journal of Applied Cryptography and serves as associate editor for ten other international journals. He is a senior member of the IEEE and a member of the IACR.



Guomin Yang graduated with a PhD in Computer Science from the City University of Hong Kong in 2009. He worked as a Research Scientist at the Temasek Laboratories of the National University of Singapore (NUS) from Sep 2009 to May 2012. He is currently a Senior Lecturer and DECRA Fellow at the School of Computing and Information Technology, University of Wollongong. His research mainly focuses on Applied Cryptography and Network Security.

Fuchun Guo received his B.S. and M.S. degrees from Fujian Normal University China in 2005 and 2008 respectively, and the PhD degree from University of Wollongong, Australia in 2013. He is currently an associate research fellow at the School of Computing and Information Technology, University of Wollongong. His primary research interest is the public-key cryptography; in particular, protocols, encryption and signature schemes, and security proof.



Xiaofen Wang received her ph.D. and M.S. degrees from Xidian University, Xian, China, in 2009 and 2006. Dr. Wang is currently a lecturer at School of Computer Science and Engineering and Big Data Research Center, University of Electronic Science and Technology of China, Chengdu, China. Her research interest are public key cryptography and its application in wireless networks, smart grid and cloud computing etc. .