Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 8, August 2017, pg.95 – 101

Smooth Projective Hash Function based PEKS Framework for Secure Searchable Cloud Storage

M. Lasya[1], NVNM Satyavani[2]

[1]M.Tech Student (CSE), 14JQ1D5815, Kakinada Institute of Technology and Science, Divili, East Godavari, Andhra Pradesh, INDIA

[2]Asst Professor, M.Tech(CSE) Department, Kakinada Institute of Technology and Science, Divili, East Godavari, Andhra Pradesh, INDIA

[1] mlasya815@gmail.com; [2] satyavani363@gmail.com

Abstract*— Public Key Encryption with Keyword Search (PEKS) idea provides a mechanism that enables users to search encrypted emails with keywords without revealing any information including the server. Also, the server can retrieve encrypted emails containing specific keywords, but learn nothing else about the emails. On the other hand, PEKS augments the security and privacy protection of data storage applications. Since cloud computing becomes the popular issue in recent years, more and more cloud services bloom in a very short time including cloud storage service. Thus, PEKS scheme can increase the personal documents protection over cloud environment. 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. 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.

***Keywords*— Public key, Dual server, PEKS, KGA, Hash function.**

I. INTRODUCTION

Considering a scenario that user Bob wants to share documents with user Alice, Bob has two options to achieve this work. Bob stores the documents in the mobile devices such as flash drive and portable hard drive, and sends it to Alice. However there are many uncertain situations in delivering process; for example, the mobile devices might be stolen by the corporate espionage so to cause huge damage to the company. Another option for Bob to share documents is taking the server as the storage media. Traditionally, users upload and store their data in the remote server and take the remote server as a storage media. Users can upload, download, update and delete the data in seconds, and they can further authorize other

users to use the data for some specific purposes. However, the security, integrity and confidentiality of data in the remote server cannot be guaranteed because users cannot control their data directly and cannot supervise how a remote server manages them clearly. The remote server is just like an untrusted third party. Therefore, users usually encrypted their documents for the privacy propose and ensuring data security before uploading to the remote server. However, as the document is transformed into a ciphertext, it produces another problem; that is how users can obtain the encrypted data without decrypting them.

Although attackers and the server administrator caies not distinguish what the context is as they capture the encrypted data, users can not define which ciphertext is the one they want, either. One of the solutions is that users download all the encrypted data and decrypt them, so that users can find the right documents they want with- out revealing any information to the server administrator. Nevertheless, this solution might cause lots of transfer cost and storage space whenever users query data. If Alice wishes to only retrieve the documents which contain the word *W*, downloading the whole encrypted data is not a suitable solution and unrealistic. Another solution is to set up keywords for each encrypted documents and a user can search the encrypted documents with specific keywords they wish to query. In order to achieve this goal, Song et al. [1] first brought up the concept of searching the encrypted data with certain words in 2000. They thought that there are two alternatives to search on the ciphertext; that is to build up an index for each word *W* and perform a sequential scan without an index. The latter do not need extra space to store the index, but slower than the former. However, the index-based schemes seem to require less sophisticated constructions, Song et al. proposed a scheme which works by computing the bitwise exclusive or (XOR) of the clear-text with a sequence of pseudorandom bits which have a special structure. The solution of Song et al. requires very little communication between the user and the server, re
quires only one round of interaction [2]. Therefore, Boneh et al. further proposed a brand-new scheme that searches the encrypted data based on keyword.

Public Key Encryption with Keyword Search (PEKS in short) scheme, which is also name searchable public-key encryption scheme, enables one to search encrypted documents on the untrusted server without revealing any information. Boneh et al. first introduced PEKS scheme with a mail routing system in 2004. There are three entities in PEKS: data sender, receiver and server. Suppose user Alice (receiver) has a number of devices: laptop, desktop, mobile device, etc. User Bob (data sender) wishes to send an email to Alice. First, he encrypts the email *M* with keywords *w*1*;w*2*; ...wm* using Alice's public key and also appends the encrypted keywords *PEKS*(*Apub;w*1), *PEKS*(*Apub;w*2), ... , *PEKS*(*Apub;wm*). Then he sends the following ciphertext to the mail server (server):

$$EApub\ (M)\ ||\ PEKS(Apub;w1)\ ||....||\ PEKS(Apub;wm)$$

Where *Apub* is Alice's public key. For Alice, she wishes to read the mails that contain keyword "urgent" using her mobile devices. For this purpose, Alice can give the server a certain trapdoor *Tw* of keyword 'urgent' that enables the server to find out the encrypted emails associated with 'urgent'. However, Alice does not want to reveal any private information to anyone including the server. In other words, the mail routing system must have the ability to test whether "urgent" is a keyword in the emails and route these mails to Alice's mobile device without getting anything else about the email. However, Boneh et al.'s scheme has to construct the secure channel to protect trapdoors throughout the transport process. This is not suitable for some applications as building a secure channel which is usually costly. To solve this problem, Baek et al. [3] proposed a new PEKS that removes the

secure channel assumption and names "Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS in short)" in 2008. In SCF-PEKS scheme, the data sender uses the server's public key and receiver's public key to encrypt the keywords each time he stores the encrypted data to the server. Whenever a receiver wants to search the encrypted data associated with a specific keyword, he can send the trapdoor to retrieve data via a *public channel*(public network) since only the server has the corresponding private key which can test whether the PEKS ciphertext matches the trapdoor. Nevertheless, the trapdoors can be transferred in the public network because trapdoors can be captured by anyone and it produces another problem: whether the outside adversaries can derive the embedded keyword and user information from the trapdoor by any means? Byun et al. [4] pointed out that PEKS might be attacked by the o®-line keyword-guessing attacks in 2006. Since keywords are chosen from much smaller space than passwords and users usually use well-known keywords (low entropy) for searching documents. Therefore, attackers can capture the trapdoor and have chance to guess keyword.

II. RELATED WORK

Nowadays, cloud storage is an important technique for sharing data, especially large media data, between enders and receivers. More and more users enjoy the services provided by cloud storage systems, such as iCloud, SkyDrive and Dropbox. Encryption is an effective and practical approach to protect the sensitive data stored in the servers.

Public key encryption with keyword search(PEKS) [5] is a notion of dealing with obtaining encrypted data from the servers with regard to cloud storages. PEKS enables a sender stores encrypted sensitive data with searchable ciphertexts into a server. A receiver who wants to obtain the sender's sensitive data should provide a keyword trapdoor to the server. After testing the validness of the keyword trapdoor with searchable ciphertexts, the server will send the encrypted sensitive data to the receiver.

The notion of PKES was first introduced by Boneh et al. [2] They proposed the first PEKS scheme, and the notion of *searchable ciphertext indistinguishability* (SC-IND) for the security of PKES. However, secure channel is required in their scheme. Baek et al. [3] pointed out that PEKS suffers from an attack that without a secure channel an attacker has the ability to link the matching trapdoor and ciphertext. Hence, they redefined the notion of SC-IND. The refined SC-IND guarantees that a malicious server that has not obtained the trapdoors for given keywords cannot tell which searchable ciphertext encrypts which keyword, and the outsider (without the server's private key) cannot make any decisions about the searchable ciphertexts even though the outsider gets all the trapdoors for the keywords that it holds.

Byun et al. [4] pointed out that PEKS can suffer from off-line keyword guessing attack: An attacker can find the keyword used for generating the trapdoor, since keywords have much lower entropy than passwords. The notion of *trapdoor indistinguishability* (TD-IND) was defined by Rhee et al. [6] to address off-line keyword guessing attack. TD-IND guarantees that without the server's private key the trapdoor does not reveal any information with respect to the keyword. Rhee et al. also proposed a new PEKS scheme which is an approach to withstand off-line keyword guessing attack. However, their dPEKS can suffer from online keyword guessing attack, which was pointed out by Yau et al. [7]. Recently, Chen [8] defined the security model for *original ciphertext indistinguishability* (OC-IND), which guarantees that the original ciphertext is indistinguishable for an outsider. They also proposed a new

framework, called secure server-designation public key encryption with keyword search (SPEKS), for withstanding online keyword guessing attack.

III.PROPOSED WORK

DS-PEKS Framework

A DS-PEKS plot primarily comprises of (KeyGen, DS-PEKS, DS-Trapdoor; FrontTest; BackTest). To be more exact, the KeyGen calculation creates general society/private key sets of the front and back servers rather than that of the collector. Besides, the trapdoor era calculation DS-Trapdoor characterized here is open while in the customary PEKS definition, the calculation Trapdoor takes as info the collector's private key. Such a distinction is expected to the diverse structures utilized by the two frameworks. In the customary PEKS, since there is just a single server, if the trapdoor era calculation is open, then the server can dispatch a speculating assault against a catchphrase ciphertext to recoup the scrambled catchphrase. Subsequently, it is difficult to accomplish the semantic security as characterized in [5]. Be that as it may, as we will appear later, under the DS-PEKS system, we can in any case accomplish semantic security when the trapdoor era calculation is open. Another distinction between the customary PEKS and our DS-PEKS is that the test calculation is isolated into two calculations, FrontTest and BackTest keep running by two free servers. This is basic for accomplishing security against within watchword speculating assault.

In the DS-PEKS framework, after getting a question from the collector, the front server pre-forms the trapdoor what not the PEKS cipher texts utilizing its private key, and afterward sends some inside testing-states to the back server with the comparing trapdoor and PEKS cipher texts covered up. The back server can then choose which reports are questioned by the collector utilizing its private key and the got inside testing-states from the front server. To begin with, Setup is performed and the framework parameters are created. Based on the created framework parameters, alternate systems are executed. It is portrayed underneath:

(1) Setup(1λ). Takes as input the security parameter λ, generates the system parameters P;

(2) KeyGen(P). Takes as input the systems parameters P, outputs the public/secret key pairs (pkFS, skFS), and (pkBS, skBS) for the front server, and the back server respectively;

(3) DS − PEKS(P, pkFS, pkBS, kw1). Takes as input P, the front server’s public key pkFS, the back server’s public key pkBS and the keyword kw1, outputs the PEKS ciphertext CTkw1 of kw1;

(4) DS − Trapdoor(P, pkFS, pkBS, kw2). Takes as input P, the front server’s public key pkFS, the back server’s public key pkBS and the keyword kw2, outputs the trapdoor Tkw2 ;

(5) FrontTest(P, skFS,CTkw1 , Tkw2). Takes as input P, the front server’s secret key skFS, the PEKS ciphertext CTkw1 and the trapdoor Tkw2 , outputs the internal testing-state CIT S;

(6) BackTest(P, skBS,CIT S). Takes as input P, the back server’s secret key skBS and the internal testing-state CIT S, outputs the testing result 0 or 1

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). As illustrated in Fig. 1, an SPHF is defined based on a domain X and an *NP* language L, where L contains a subset of the elements of the domain X, i.e., $L \subset X$. Formally, an SPHF system over a language $L \subset X$, onto a set Y, is defined by the following five algorithms (SPHFSetup, HashKG, ProjKG, Hash, ProjHash):

(1) SPHFSetup(1λ): generates the global parameters param and the description of an *NP* language instance L;

(2) HashKG$(L,$ param$)$: generates a hashing key hk for L;

(3) ProjKG$($hk, $(L,$ param$))$: derives the projection key hp from the hashing key hk;

(4) Hash$($hk, $(L,$ param$),W)$: outputs the hash value hv $\in Y$ for the word W from the hashing key hk;

(5) ProjHash$($hp, $(L,$ param$),W,w)$: outputs the hash value hv $\in Y$ for the word W from the projection key hp and the witness w for the fact that $W \in L$.

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

Hash$($hk, $(L,$ param$),W)$ = ProjHash$($hp, $(L,$ param$), W,w)$.

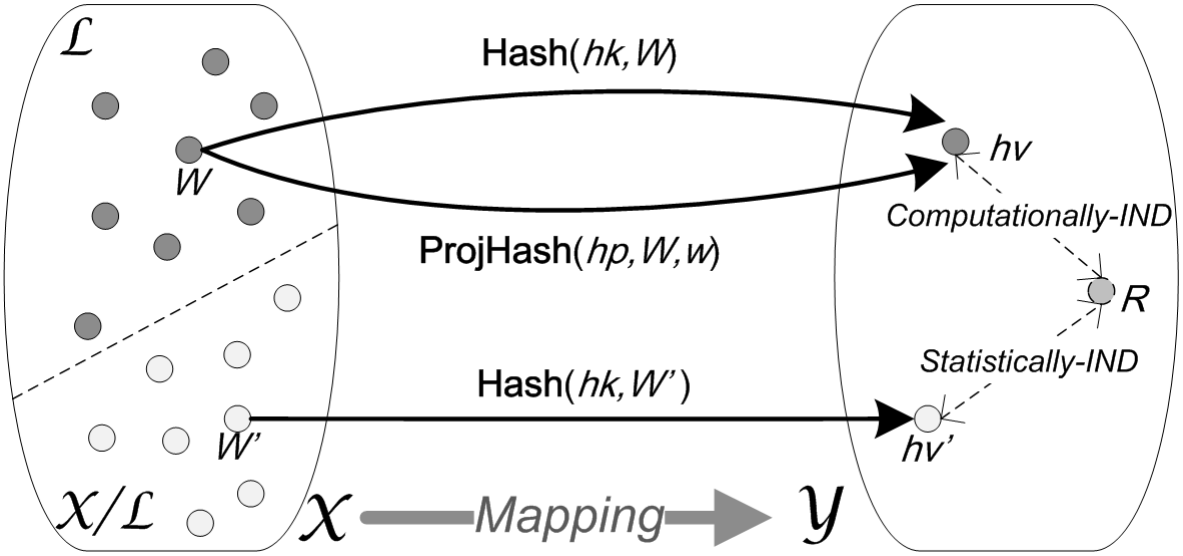

Figure 1. Smooth projective hash function.

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

$V1 = \{(L,$ param$,W,$ hp,hv$)|$hv=Hash$($hk, $(L,$ param$),W)\}$,
$V2 = \{(L,$ param$,W,$ hp,hv$)|$hv $\$\leftarrow Y\}$,

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

An instantiation of DS-PEKS Framework

A linear and homomorphic language *LDH* can be derived from the Diffie-Hellman assumption and then construct a concrete linear and homomorphic SPHF, referred to as *SPHFDH*, from *LDH*. We provide a formal proof that *SPHFDH* 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 *SPHFDH*. . 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. 1 and Fig. 2 to highlight their key properties.

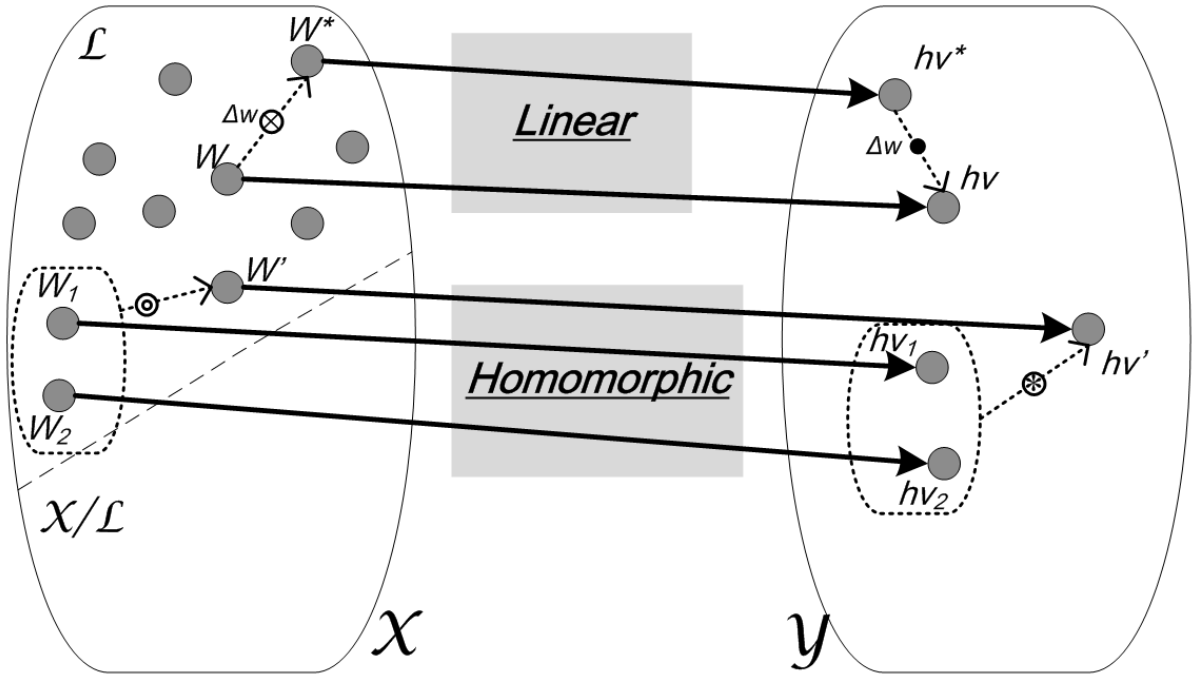

Figure 2. Linear and Homomorphic SPHF.

We introduce the Diffie Hellman language *LDH* and then derived an LH-SPHF. Let G be a group of primer order *p* and *g*1, *g*2 ∈ G the generators of G.

1) Decision Diffie-Hellman (DDH) Assumption: The DDH assumption says that for any *a*, *b*, *r* randomly chosen from Z*p*, {(*g*1, *ga*1 , *gb*1, *gab*1)} is computationally indistinguishable from {(*g*1, *ga*1 , *gb*1, *gr*1)}.

2) Diffie-Hellman Language: The Diffie-Hellman Language is defined as follows.

$$LDH = \{(u1, u2) | \exists\, r \in Zp, s.t., u1 = gr1, u2 = gr2\}$$

IV. CONCLUSIONS

Now a day's there will be growing popularity of cloud computing, large number of users and data owners are motivated to outsource their data to cloud servers for large convenience and reduced cost required for data management. However, important data should be encrypted before outsourcing for privacy requirements, which uses data utilization technique like keyword-based document retrieval. Searchable encryption is of expanding enthusiasm for ensuring the information protection in secure searchable distributed storage. In this paper, we research the security of a notable cryptographic primitive, in particular, public key encryption with keyword search (PEKS) which is exceptionally valuable in numerous utilizations of cloud storage. The conventional PEKS structure experiences an inalienable instability called inside Brut force keyword guessing attack propelled by the pernicious server. To address this security defenselessness, we propose another PEKS system named double server PEKS (DSPEKS). As another principle commitment, we characterize another variation of the smooth projective hash capacities (SPHFs) alluded to as direct and homomorphic SPHF (LH-SPHF). We then demonstrate a bland development of secure DS-PEKS from LH-SPHF.

REFERENCES

[1] D. W. Dawn, X. Song and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S P 2000. Proceedings. 2000IEEE Symposium on*, pp. 44 {55, 2000.

[2] D. Boneh, G. D. Crescenzom, R. Ostrovsky, and G. Rersiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science*, vol. 3027, pp. 506{522, Interlaken, Switzerland, 2004. Springer Berlin/Heidelberg.

[3] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *ICCSA 2008*, vol. 5072 of *Lecture Notes in Computer Science*, pp. 1249{1259, Perugia, Italy, 2008. Springer Berlin/Heidelberg.

[4] J. W. Byun, H. A Park, H. S. Rhee, and D. H. Lee, "Online keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management, Lecture Notes in Computer Science*, vol. 4165, pp. 75{83, Seoul, Korea, 2006. Springer Berlin/Heidelberg.

[5] Abdalla, M. et al.: 'Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions'. J. Cryptol., 21, 2008. pp. 350-391.

[6] Rhee, H.S., Park, J.H., Susilo,W., Lee, D.H.:'Trapdoor security in a searchable publickey encryption scheme with a designated tester'. J. Syst. Softw., 83, 2010. pp.763-771.

[7] Yau,W.C., Phan, R.C., Heng, S.H., Goi, B.M.:'Keyword Guessing Attacks on Secure Searchable Public Key Encryption Schemes with a Designated Tester'. Int. J. Comput. Math., 90, 2013. pp.2581-2587.

[8] Chen, Y.C.: 'SPEKS: Secure Server-Designation Public Key Encryption with Keyword Search against Keyword Guessing Attacks'. The Computer Journal, Vol. 58 No. 4, 2015. pp.922-933.

