Information Sciences 403–404 (2017) 1–14

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks

Qiong Huang[a,*], Hongbo Li[b]

[a] College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China
[b] Nanjing University of Information Science & Technology, Nanjing, China

ARTICLE INFO

Article history:
Received 19 December 2016
Revised 18 February 2017
Accepted 31 March 2017
Available online 1 April 2017

Keywords:
Public key authenticated encryption with keyword search
Searchable encryption
Keyword guessing attack
Random oracle model

ABSTRACT

How to efficiently search over encrypted data is an important and interesting problem in the cloud era. To solve it, Boneh et al. introduced the notion of public key encryption with keyword search (PEKS), in 2004. However, in almost all the PEKS schemes an inside adversary may recover the keyword from a given trapdoor by exhaustively guessing the keywords offline. How to resist the inside keyword guessing attack in PEKS remains a hard problem.

In this paper we propose introduce the notion of *Public-key Authenticated Encryption with Keyword Search* (PAEKS) to solve the problem, in which the data sender not only encrypts a keyword, but also authenticates it, so that a verifier would be convinced that the encrypted keyword can only be generated by the sender. We propose a concrete and efficient construction of PAEKS, and prove its security based on simple and static assumptions in the random oracle model under the given security models. Experimental results show that our scheme enjoys a comparable efficiency with Boneh et al.'s scheme.

1. Introduction

With the rapid development of cloud computing technology, a large amount of data now has been stored onto the cloud. Since the data owner loses its control of the data, several security and privacy issues arise in cloud storage service, among which data privacy is a very sensitive problem. Encryption is an effective way to protect the data from being leaked. However, traditional search mechanisms do not work for encrypted data. How to efficiently search over encrypted data thus becomes an important and interesting problem, and has attracted many researchers' attention [1–12].

In 2004, Boneh et al. introduced the notion of Public Key Encryption with Keyword Search (PEKS), integrating keyword search functionality into public key encryption. They proposed the first PEKS scheme (denoted by BDOP-PEKS hereafter) [1] and showed that it is secure based on the bilinear Diffie-Hellman assumption in the random oracle model. The framework of PEKS is illustrated in Fig. 1.

There are three parties involved, a data sender called Alice, a data receiver called Bob and a cloud server. Alice has a bunch of sensitive documents $\{F_i\}$ to share with his friend Bob. First, Alice extracts keywords $\{w_{i,j}\}$ from each document F_i, and encrypts the keywords using the PEKS scheme. Besides, Alice encrypts each document with a (possibly another) encryption scheme. Let the ciphertexts be $\{C_{w_{i,j}}\}$ and $\{C_i\}$, respectively. Alice uploads all the ciphertexts onto the cloud server. To search over the encrypted documents whether there is any one containing some keyword w, Bob computes a

* Corresponding author.
E-mail addresses: qhuang@scau.edu.cn (Q. Huang), hongbo@stu.scau.edu.cn (H. Li).

http://dx.doi.org/10.1016/j.ins.2017.03.038
0020-0255/

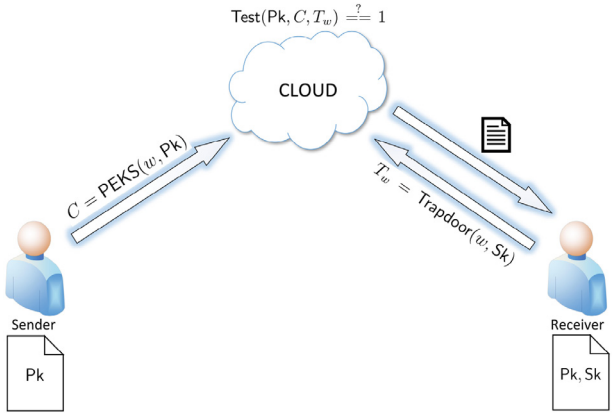

Fig. 1. PEKS System Framework.

trapdoor T_w for w using his secret key, and gives it to the cloud server (via a secure channel). With T_w, the server runs the Test algorithm to test each encrypted keyword $C_{w_{i,j}}$ weather it contains the same keyword as T_w or not. If $C_{w_{i,j}}$ matches T_w, the associated document contains w. When the search finishes, the server returns the search result to Bob. Notice that during the search, the server does not know the content of the documents, nor the keyword.

1.1. Keyword guessing attack

Ideally, the keyword space is assumed to be at least super-polynomially large. However, in real applications it is usually not that large. Keywords are often chosen from a low-entropy keywords space. Therefore, it may be feasible for the adversary to guess what keywords a document contains by launching the *keyword guessing attack* (KGA) [2,3]. Roughly speaking, in a KGA attack the adversary tries each possible keyword, encrypts it, and tests the ciphertext with the given trapdoor. If the test succeeds, the adversary knows which keyword is encapsulated in the given trapdoor. Because people usually choose keywords which are used frequently and easy to memorize, the server is able to find out the underling keyword. Assuming in an encrypted mail system, user Alice sends an encrypted email attached with the PEKS ciphertext of a keyword to another user Bob. An adversary can apply the KGA attack on the PEKS ciphertext and may reveal the keyword of the email if the trapdoor sent from Bob to the email server is available. In this way, the adversary may know the theme of the email and thus the user's privacy is leaked. Such an attack is usually launched by the cloud server or any other role inside the cloud service management. Therefore, it is also called *Inside Keyword Guessing Attack* (IKGA).

To resist the KGA attacks, researchers have proposed several ideas and introduced different notions. Roughly speaking, the KGA works for two reasons possibly. First, the adversary could get the trapdoor. Second, it can do the test freely. Therefore, in order to prevent an adversary from launching a KGA attack, one can either protect the trapdoor from being leaked to an outside attacker, for example, setting up a secure channel between the receiver and the server so that only the server can get the trapdoor; or restrict the unauthorized adversary from doing the test, i.e. *designated-tester* PEKS [7,8] (i.e. only the designated server can do the test), PEKS with *authorization* [4,5] (i.e. only the authorized one can do the test). However, neither method can prevent an inside adversary from launching the KGA attack. Hence, how to build a (public-key) searchable encryption scheme which is secure against inside keyword guessing attack is still an open problem.

1.2. Related works

Recently, many PEKS schemes and variants have been proposed in the literature. Roughly speaking, these works can be classified into the following types: 1) multi-users access control in PEKS [4,5,13], 2) fuzzy keyword search in PEKS [14–16], 3) flexible keyword search in PEKS [17–19], and 4) trapdoor privacy in PEKS [2,3,20–22]. Works on trapdoor privacy mainly focus on preventing the adversary from revealing the keywords from given ciphertexts. However, to the best of our knowledge, almost all of them can not resist the inside keyword guessing attack. The only scheme [23] known to counterattack the IKGA attack makes use of two servers and assumes that the two servers do not collude.

There are several ways to leak information about the keyword. For example, the search result, e.g. the number of records satisfying the search query, leaks certain information about the keyword in the trapdoor, which seems to be unavoidable if the server is curious about the keyword. Besides, the trapdoor itself might reveal certain information about the keyword [20]. In existing PEKS schemes, e.g [1,24], keywords may be revealed due to the leakage of search patterns [21]. The server is often assumed to be honest-but-curious. It would honestly do its job but try to find more information about the keywords (as well as the document contents). The server has the capability to monitor the communication channels and obtain all encrypted data, indices and trapdoors. Once the server reveals users' search pattern, the search frequency of a keyword would be exposed to the server, leading to privacy leakage.