An Efficient Symmetric Searchable Encryption Scheme for Cloud Storage

Xiuxiu Jiang[1, 2], Xinrui Ge[1], Jia Yu[1, 2, 3*], Fanyu Kong[4], Xiangguo Cheng[1], and Rong Hao[1]
[1]College of Computer Science and Technology, Qingdao University, 266071 Qingdao, China
[2]Institute of Big Data Technology and Smart City, Qingdao University, 266071 Qingdao, China
[3]State Key Laboratory of Information Security, Institute of Information Engineering
Chinese Academy of Sciences, 100093 Beijing, China
[4]Institute of Network Security, Shandong University, 250100 Jinan, China

Abstract

Symmetric searchable encryption for cloud storage enables users to retrieve the documents they want in a privacy-preserving way, which has become a hotspot of research. In this paper, we propose an efficient keyword search scheme over encrypted cloud data. We firstly adopt a structure named as inverted matrix (IM) to build search index. The IM is consisted of index vectors, each of which is associated with a keyword. Then we map a keyword to an address used to locate the corresponding index vector. Finally, we mask index vectors with pseudo-random bits to obtain an encrypted enlarged inverted matrix (EEIM). Through the security analysis and experimental evaluation, we demonstrate the privacy and efficiency of our scheme respectively. In addition, we further consider two extended practical search situations, i.e., occurrence queries and dynamic user management, and then give two relevant schemes.

Keywords: symmetric searchable encryption, cloud storage, privacy, efficiency

1 Introduction

Cloud computing has become a new ideal model of calculation [11, 13, 29]. It has been considered as the next generation and extension of information technology. It has great benefits for consumers, including on-demand high quality services, ubiquitous network access, rapid configuration of computing resources and flexible charging methods, etc. The new promising computing model[22] can not only deal with the pressure on individual storage management, but also avoid devoting massive resources to the local hardware and software for users. Owing to the advantages of cloud computing, both enterprises and individuals storage their local large amounts of data on the cloud actively.

However, cloud service providers (CSP) are independent entities. The internal operation details of cloud platform are opaque to users. In the cloud storage system, users cannot possess their data physically any more causing that users and cloud service providers are not in the same trusted domain. So users cannot fully trust the cloud service provider. With full access to the underlying operating system, the cloud administrators may actively reveal data information to unauthorized parties. Moreover, the server may be subject to attacks from hackers, leading to information disclosure passively. For the sake of protecting the privacy of the data, it is necessary to encrypt the sensitive data, such as e-mails, personal medical records, company secrets, government documents, from consumers, before being uploaded to the cloud. Considering the large amount of outsourcing data and data sharing in the cloud computing, data owners want to receive the data of interest each time accessing to the cloud server. So for users, there are new challenges to search the encrypted cloud data[14]. The simplest way of solving this problem is

Journal of Internet Services and Information Security (JISIS), volume: 7, number: 2 (May 2017), pp. 1-18
*Corresponding author: Email: qduyujia@gmail.com, Tel: +86053285953215

downloads all the cipher-text. Then users decrypt all the data locally. As we all know, this method is obviously impractical, because it needs huge amount of bandwidth cost in cloud scale systems[1, 6]. Besides, keyword-search approaches allow users to obtain their favourite files. These approaches also have been extensively used in the plain-text search context, such as Google search. But unfortunately, these traditional keyword-search approaches regarding plaintext aren't applicable to encrypted cloud data directly by reason that the data encryption complicates data utilization services. Hence, the most helpful method to address the problem is searchable encryption. It allows users to selectively retrieve files from the cloud using a keyword-based search technology [18].

In recent years, the searchable encryption [4, 8, 9, 15, 21, 23] has been studied intensively by a growing number of researchers. Various searching keywords schemes on the encrypted cloud data have been put forward. These programs generally create an encrypted index which will not reveal its contents until the server is given a valid trapdoor. Searchable encryption can be divided into two main groups: public key encryption with keyword search and searchable symmetric encryption. Boneh et al [4] proposed the first searchable encryption scheme. This scheme is used in the public-key cryptography field, where anyone with the recipient's public key can create searchable contents, but only when the user holds private key can generate valid query requests and decrypt the encrypted documents returned from the cloud. However, although public key searchable encryption search can support for more application scenarios, these constructions commonly use a bilinear map, resulting in a large computation cost and low search efficiency. Compared with the search schemes based on public key, searchable symmetric encryption with important features such as less calculation, easy to implement, fast computational speed is more practical in the real life. Song et al[9] proposed the first searchable symmetric encryption scheme. It is using the block cipher technology in the symmetric key setting. Due to searching the whole cipher-texts with a sequential scan, it spends amount of searching overhead. Goh et al[15] first formulated a security index model. It is defined as a semantic security against adaptive chosen keyword attacks (IND1-CKA)[15]. IND1-CKA cannot guarantee the trapdoor privacy. They also constructed a IND1-CKA secure index using a Bloom filter and applied it to search over encrypted data. Curtmola et al [23] gave two formal security notions, i.e. non-adaptive indistinguishability against chosen-keyword attack (IND-CKA1) and adaptive indistinguishability against chosen-keyword attack (IND-CKA2), presented two efficient constructions with improved security definitions based on an inverted index data structure and first defined searchable encrypted encryption in the multi-user setting. In their construction, they made use of the inverted index to build a hash index table for the whole document collection. Chang et al.[8] introduced two practical schemes. But in their first scheme, users need to store and manage a prebuilt dictionary locally. Although this problem was solved in the second scheme, it requires two interactions between users and the server, which will undoubtedly degrade the user experience. Chase et al. [21] proposed an adaptively secure construction which generates a padded and permuted dictionary as an inverted index. The scheme is IND-CKA2 secure and hides the data structure as well. In order to satisfy users' increasing needs and enrich search functionality, some new designs supporting conjunctive keyword search [3, 5, 10, 16, 17] ranked search [26–28, 30] and secure dynamic update[7, 12, 20, 24] have been developed in recent years.

In this paper, we propose an efficient symmetric searchable encryption scheme for cloud storage. We firstly use a structure named as Inverted Matrix (IM) to create search index. The IM is made up of a number of index vectors which are subindices for distinct words in the data set. Specifically, while constructing the index, each keyword is associated with an index vector and the index vector is denoted by binary bits. Each bit represents if the keyword appears in the related document. Then we map a keyword to a value as an address used to locate the corresponding index vector. As a result, users can avoid the overhead of storing a dictionary locally. Finally, to preserve users' privacy, we blind the index vectors using pseudo-random bits to obtain an Encrypted Enlarged Inverted Matrix (EEIM) which can prevent the server from learning information from the index. Therefore, when a search query is submitted, the

server just needs to find the matching index vector via the inputted valid request. Users only need a single interaction with the server to get the data files they are interested in. The major contributions of this paper are as follows:

1. Based on the new index structure IM, we propose an efficient symmetric searchable encryption scheme.

2. Through the security analysis and experimental evaluation, we show that the proposed scheme is adaptive semantic secure and efficient.

3. Upon the proposed search index structure, we present an extended solution that makes our construction can deal with occurrence search to enhance our utility of our scheme in practice.

4. By introducing a dynamic user management mechanism, we make our scheme applicable to multi-user scenario.

Compared with the conference version in 3PGCIC [19], this paper has the following main differences. Firstly, we give a more complete and comprehensive introduction for the related work of our proposed scheme. Secondly, we describe how to extend our scheme to handle the occurrence queries like "I want to search all the documents each of which contains 'Panda' at least ten times". Furthermore, we consider to realize dynamic user management which makes it possible that multiple authorized data users are capable of submitting search requests and sharing the outsourcing data with the data owner. Finally, with the extensive experimental evaluation results and detailed performance analysis, we verify the efficiency of our raised construction.

The rest of this paper is organized as follows. In section II, we introduce the problem formulation, mainly including system model, threat model and design goals. Then section III describes preliminaries, the definition of Algorithms, and provides the details of our proposed scheme. Followed by Section IV and section V, they analyze security and performance evaluation. In section VI, we consider two useful extensions to our basic searchable encryption scheme. In the end, section VII give a conclusion of this paper.

2 Problem Formulation

2.1 System Model

In this paper, we construct the cloud system model as Fig.1. It involves three different kinds of parties: the data owner, the data user and the cloud server. The data owner has a set of documents $\mathscr{D}$ and wishes to outsource it in an encrypted form $\mathscr{C}$ to the cloud server. For the sake of search efficiency and data privacy, before uploading, the data owner will firstly extract a set of keywords Δ from plain-texts and then construct a secure encrypted index $\mathscr{I}$ for the whole data set. The encrypted index $\mathscr{I}$ along with cipher-texts $\mathscr{C}$ will be uploaded to the cloud server. In cloud computing, we suppose that it is well done to authorize users for data owners. When the data users want to retrieve the files containing a specific keyword of interest, they need to send a corresponding valid trapdoor T_w to the cloud server to tell cloud that they want to search for some related files. After receiving T_w from a data user, the cloud server will execute the search operation over the secret index $\mathscr{I}$ without any decryption action. The cloud will return all the related cipher texts containing the queried keyword [6]. Finally, data users locally decrypt the received cipher-texts with a secret key. The problem of user management in our model will be introduced as an extension in section VII. The data owner will transport the trapdoors and data decryption to the data users in a secure channel.

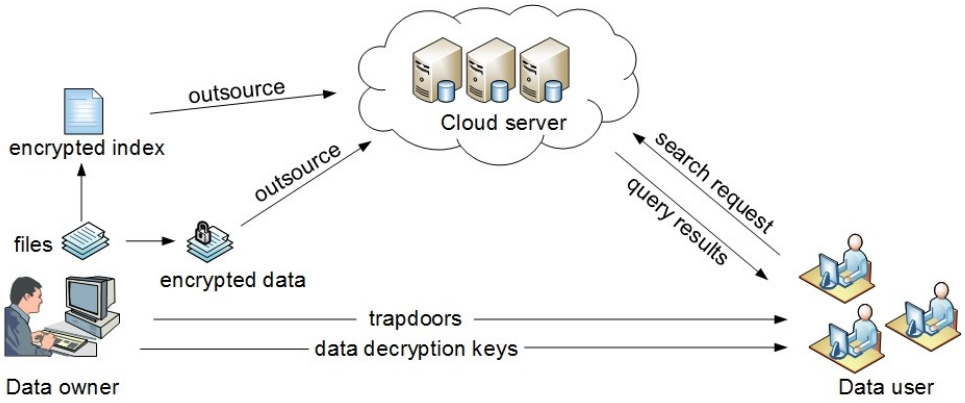


Figure 1: **Architecture of search over encrypted cloud data**

2.2 Thread Model

In this model, we consider the cloud server as "honest-but-curious", as mentioned in most of the works on cloud security[23]. Its "honesty" is shown in that:

1. the cloud server won't destroy and temper the stored data;

2. the cloud server will honestly act in pre-defined protocol, i.e., perform the search operation on the encrypted data for a given keyword and send back the corresponding files associated with the queries. Its "curiosity" lies on that the cloud server tries to learn the rest of information from the search queries and the index[6].

2.3 Design Goals

In this paper, our scheme should satisfy the following requirements:

1. *Keywordsearch*. The scheme allows data users to submit a search query for a keyword and retrieve all the documents containing the keyword.

2. *Privacypreserving*. It means that nothing is leaked to cloud server beyond access pattern (i.e., the ids of the files containing a keyword) and search pattern(i.e., revealing if every search is for the same word or not). We consider three representative privacy guarantees as follows.

   1) *Documentprivacy*. This scheme must assure that adversaries won't know anything from the encrypted documents. They might only know the respective size, length and ids.

   2) *Indexprivacy*. It requires that the cloud server could not deduce any associated information between keywords and the stored documents from the encrypted index.

   3) *Trapdoorprivacy*. It guarantees that the trapdoors will not leak any information about the queried words.

3. $Efficiency$. Our scheme should be accomplished with low computation overhead. The running time of the algorithms in the scheme is applicable in practice.

3 The Proposed Scheme

In this section, we will present the main algorithms of our scheme and show how to make use of the Inverted Matrix (IM) to achieve a complete efficient keyword search over encrypted data in the cloud storage environment step by step. We begin with introducing some notations and relevant structures which are used in the constructions.

3.1 Preliminaries

3.1.1 Notation

For easier reading, some of the symbols used in this paper are stated as follows.

- Let $\Delta = (w_1, w_2, ..., w_m)$ be a dictionary consisting of m distinct keywords.
- Let $\mathscr{D} = (D_1, D_2, ..., D_n)$ be a collection of n plain-text documents.
- Let $\mathscr{C} = (C_1, C_2, ..., C_n)$ be a set of n encrypted files stored in the cloud.
- Let $id(D)$ be the id of document D.
- Let $D(w)$ denotes the set of the all files' ids in $\mathscr{D}$ containing the keyword w.
- Let T_w be the trapdoor as a search request for the keyword w.
- Let $\mathscr{I}$ be the index constructed by data owner for the whole document collection.

3.1.2 IM,EIM,EEIM

IM(Inverted Matrix)

We suppose that IM is a matrix composed of m lines and n columns, where m is the number of distinct keywords in the dictionary Δ and the number of documents is n in the data collection $\mathscr{D}$, as shown in Fig.2. Each line in IM is a binary vector corresponding with a keyword in Δ, which expresses the containment relationships between the keyword and every document in Δ. For example, we express the element at the i-th line and the j-th column in IM as $IM_{i,j}$, where $1 \leq i \leq m$,$1 \leq j \leq n$. If the value of $IM_{i,j}$ is 1, it means that the document D_j contains the keyword w_i. Otherwise, it means the document D_j doesn't contain the keyword w_i. Given any word w , we can search documents which have the word w via its vector.

EIM(Enlarged Inverted Matrix)

EIM is an enlarged inverted matrix based on IM. To be specific, IM is "stretched"to generate EIM, while remaining unchanged in width, as illustrated in Fig.2. The main difference between them is that EIM has much more lines than IM. We suppose that keywords in Δ can be represented by up to d bits. Firstly, we initiate the EIM by padding 2^d zero vectors which are considered as "empty". Then we map each vector in IM to an empty line in EIM according to a certain rule, such that the two vectors in corresponding places are the same. Without loss of generality, we generally adopt a pseudo-random permutation with taking a keyword as input to serve as the rule, ensuring the users can locate the matching vector via the word they want to search.

EEIM(Encrypted Enlarged Inverted Matrix)

EEIM is actually an encryption of EIM. We employ the method of masking the vectors of EIM to hide the real information. It seems to the cloud server that the entries in EEIM store arbitrary binary strings with no rules. The final encrypted index constructed in our scheme is EEIM.

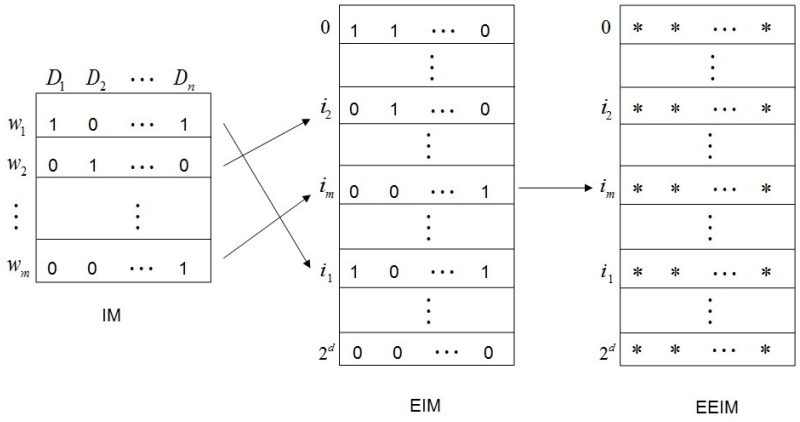


Figure 2: **An example of the three structures**

3.2 The Definition of Algorithm

There are four polynomial-time algorithms (Keygen, BuildIndex, Trapdoor, Search) in our scheme, such that,

1. Keygen(1^k) is used to generate a key.It is a probabilistic algorithm. Inputing a security parameter k and outputing a secret key K.

2. BuildIndex$(K, \mathscr{D})$ is an indexes construction algorithm. It inputs a secret key K and a document collection $\mathscr{D}$, and outputs an index $\mathscr{I}$.

3. Trapdoor(K, w) is a trapdoor generation algorithm. User inputs a secret key K and a keyword w, and it outputs a corresponding trapdoor T_w.

4. Search$(\mathscr{I}, T_w)$ is a search algorithm. It is took by the cloud. The cloud inputs the encrypted index $\mathscr{I}$ for a data collection $\mathscr{D}$ and a trapdoor T_w , and the algorithm outputs a set of documents ids $D(w)$.

3.3 The Proposed Scheme

In our construction, we assume that the length of words in the dictionary Δ is at most d bits. We find a connection between a single index $\mathscr{I}$ and a document collection $\mathscr{D}$ and associate them. The index $\mathscr{I}$ is denoted by an array of size 2^d. Recall that the size of Δ is m and the size of $\mathscr{D}$ is n. Also, we assume that the documents in $\mathscr{D}$ is ordered in lexicographic order of their identifiers and there is a specific array storing the identifiers of the documents, so it is available to find $id(D_i)$ for the server. Besides, we make use of a pseudo-random permutation π, a pseudo-random function f with the following parameters:

$$\pi : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^d$$

$$f : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$$

Our proposed complete scheme is as follows.

1. Keygen(1^k).

   The data owner chooses a security parameter k , generates random keys $K_1, K_2 \in \{0,1\}^k$ and outputs the secret key $K = (K_1, K_2)$.

2. BuildIndex($K, \mathscr{D}$).

   The process of building index is as follows.

   (1) The data owner selects the different keywords in all the files in the set $\mathscr{D}$ to generate the dictionary Δ. For each word $w_i \in \Delta$, he/she builds $D(w_i)$.

   (2) The data owner then builds up an IM-based index structure. He/she generates a n-bit binary index vector I_i for each keyword $w_i \in \Delta$. Each binary bit $I_i[j]$ symbols if the related document D_j contains the keyword w_i. In other words, for each file D_j,$1 \leq j \leq n$, if $id(D_j) \in D(w_i)$, then he/she sets $I_i[j] = 1$; otherwise, sets $I_i[j] = 0$.

   (3) Let $\mathscr{I}$ be an array of size 2^d. For each word $w_i \in \Delta$, the data owner computes $\pi_{K_1}(w_i)$ with the random key K_1 and then sets $\mathscr{I}[\pi_{K_1}(w_i)] = I_i$. Other $(2^d - m)$ entries of $\mathscr{I}$ are set to zero vectors with n-bit size.

   (4) For all $addr = 1, ..., 2^d$, the data owner computes $f_{K_2}(addr)$ with the random key K_2 to obfuscate the original information. For $i = 1, 2, ..., m$, if $addr = \pi_{K_1}(w)$, then he/she sets $\mathscr{I}(addr) = I_i \oplus f_{K_2}(addr)$; otherwise, sets $\mathscr{I}(addr) = 0^n \oplus f_{K_2}(addr)$.

   (5) The data owner outputs the index $\mathscr{I}$.

3. Trapdoor(K, w).

   With a keyword of interest w , data users compute $\pi_{K_1}(w_i)$ and $f_{K_2}(\pi_{K_1}(w))$ using the secret keys K_1, K_2 , then send we see $T_w = (\pi_{K_1}(w), f_{K_2}(\pi_{K_1}(w)))$ as a search trapdoor and send it to the cloud.

4. Search($\mathscr{I}, T_w$).

   (1) Parse T_w as (α, β). As soon as the cloud receive the T_w , it firstly locates the matching index vector via α , denoted as γ, and then uses β to decrypt the vector γ. Let θ be the unencrypted index vector, i.e. $\beta \oplus \gamma = \theta$. If $\theta = 0^n$, it means that there are no documents containing the inputted keyword and the server needs to response a failure message to the users.

   (2) Otherwise, the cloud server scans every bit of the index vector θ and finds which documents contain the keyword being required. Let X be an empty set. For $j = 1, ...n$, the cloud server adds $id(C_j)$ to X if $\theta[j] = 1$. At last, the cloud server fetches the documents according to the identifiers in X and sends back them to the users.

4 Security Analysis

Similar to [23], we formalize a simulation-based adaptive semantic security definition that meets three privacy requirements described in design goals. Then, we demonstrate that our proposed scheme is secure in the adaptive setting.

For actually, all the encrypted documents are stored at the cloud server, the query interactions between the client and server will leak access pattern and search pattern inevitably. Thus, like most previous secure searchable encryption schemes [23],[16], [7, 12, 24, 28], our work will not take the privacy information as what we should protect. To express the security analysis briefly, with following a similar leakage profile of [24], we define two specified leakage functions L_1 and L_2 to describe what will be

deduced from the view of the attacker. The function $L_1(\mathscr{I},\mathscr{D})$, its variable is the index $\mathscr{I}$ and the document collection $\mathscr{D}$ and outputs the total size of the dataset $|\mathscr{D}|$, the number of files m , the length of every document $|D_j|$ and the identifiers of each document $id(D_i)$. The function $L_2(\mathscr{I},\mathscr{D},\mathscr{Q})$ inputs the index $\mathscr{I}$, $\mathscr{D}$ and the keywords set$\mathscr{Q}$ that we searched in the past and reveals the access pattern search patterns. As our thought is independent of the encryption method that we choosed for the outsourcing data, the plaintext can be encrypted by a well-known semantically secure symmetric encryption algorithm separately. Let ε_{sk} be PCPA-secure symmetric encryption scheme.

Definition 1 (**adaptive semantic security**). *Let L_1 and L_2 be stateful leakage functions, $\mathscr{A}$ be a stateful adversary and $\mathscr{S}$ be a stateful simulator. Let $k \in N$ be the security parameter. We take the two probabilistic experiments $\textbf{\textit{Real}}_{\mathscr{A}}(k)$ and $\textbf{\textit{Sim}}_{\mathscr{A},\mathscr{S}}(k)$ into consideration. Of which, the challenger and the adversary $\mathscr{A}$ implement the $\textbf{\textit{Real}}_{\mathscr{A}}(k)$, and $\textbf{\textit{Sim}}_{\mathscr{A},\mathscr{S}}(k)$ is carried by $\mathscr{A}$ and $\mathscr{S}$.*

$\textbf{Real}_{\mathscr{A}}(k)$: The challenger generates a key K by running $Keygen(1^k)$ and randomly chooses a secret key sk for ε. $\mathscr{A}$ sends $\mathscr{D}$ to the challenger. The challenger returns $\mathscr{I} \leftarrow \text{BuildIndex}(K,\mathscr{D})$ and $\mathscr{C} \leftarrow \varepsilon_{sk}(\mathscr{D})$ to $\mathscr{A}$. The adversary submits q queries, where q is polynomial in k . The set of q keywords is denoted as $Q = (w_1, w_2, ..., w_q)$. For each keyword, the adversary receives a trapdoor $T_w \leftarrow \text{Trapdoor}(K,w)$ from the challenger and picks the next keyword as a function of previously obtained trapdoors and search outcomes. Finally, $\mathscr{A}$ outputs a bit b .

$\textbf{Sim}_{\mathscr{A},\mathscr{S}}(k)$: $\mathscr{A}$ chooses a document collection $\mathscr{D}$ and sends it to $\mathscr{S}$. Given $L_1(\mathscr{I},\mathscr{D})$, $\mathscr{S}$ computes $(\mathscr{I}',\mathscr{C}')$ and sends them to $\mathscr{A}$. In the search phase, $\mathscr{A}$ makes q queries expressed as Q . For each keyword, $\mathscr{S}$ learns from $L_2(\mathscr{I},\mathscr{D},\mathscr{Q})$ and returns T'_w to $\mathscr{A}$. Finally, $\mathscr{A}$ outputs a bit b.

We say that our scheme satisfies adaptive semantic security if for all probabilistic polynomial-time (PPT) adversaries $\mathscr{A}$, there exists a PPT simulator $\mathscr{S}$ such that $|Pr[\textbf{Real}_{\mathscr{A}}(k) = 1]| - |Pr[\textbf{Sim}_{\mathscr{A},\mathscr{S}}(k) = 1]| \leq negl(k)$, where $negl(k)$ is a negligible function.

Proof Let $\mathscr{A}$ and $\mathscr{S}$ be an adversary and a simulator in definition 1 respectively. We construct a PPT $\mathscr{S}$ such that for all PPT $\mathscr{A}$, the advantage to distinguish the outputs of above experiments $\textbf{Real}_{\mathscr{A}}(k)$ and $\textbf{Sim}_{\mathscr{A},\mathscr{S}}(k)$ is negligible. The simulator $\mathscr{S}$ adaptively generates $(\mathscr{I}',\mathscr{C}',T'_w)$ as follows.

1. Since $\mathscr{S}$ knows the size of each documents in $\mathscr{D}$ from the leakage algorithm $L_1(\mathscr{I},\mathscr{D})$, it will simulate the encrypted documents $C'_j \leftarrow \{0,1\}^{|D_j|}$ for $j = 1,...n$. Let $\mathscr{C}' = (C'_1,...,C'_n)$. To simulate the secure index $\mathscr{I}$, $\mathscr{S}$ sets $\mathscr{I}'$ to be a $0,1^n \times 2^d$ array. For $1 \leq i \leq 2^d$, we randomly chooses $S_i \leftarrow \{0,1\}^n$ and stores it in each entry of $\mathscr{I}'$. Then $\mathscr{S}$ sends $(\mathscr{C}',\mathscr{I}')$ to $\mathscr{A}$.

2. For the first query, $\mathscr{S}$ picks an address $addr_1$ from $\mathscr{I}'$ at random after $\mathscr{A}$ sends w_1 to $\mathscr{S}$. We denotes $\mathscr{I}'[addr_1]$ as s_1 . Though the leakage function $L_2(\mathscr{I},\mathscr{D},\mathscr{Q})$, $\mathscr{S}$ learns $D(w_1)$ and generate a n -bit string I_1 . For $j = 1,..,n$, it sets $I_1[j] = 1$ if $id(D_j) \in D(w_1)$; otherwise, sets$I_1[j] = 0$. Then $\mathscr{S}$ computes $\beta_1 = s_1 \oplus I_1$ and then returns $T'_{w_1} = (addr_1, \beta_1)$ to $\mathscr{A}$.

3. For $i = 1,...,q$, first $\mathscr{S}$ checks whether w_i has appeared before. This can be known by checking the search pattern revealed by $L_2(\mathscr{I},\mathscr{D},\mathscr{Q})$. If w_i did previously appear, then $\mathscr{S}$ retrieves the trapdoor previously used for w_i and uses it as T'_{w_i} . However, if w_i has not previously appeared, then $\mathscr{S}$ generates a trapdoor T'_{w_i} in the same way as the second step, making sure that the $addr_i$ we choose is distinct from any $addr$ used previously. Finally,$\mathscr{S}$ returns $T'_{w_i} = (addr_i, \beta_i)$ to $\mathscr{A}$.

Recall that C_i is ε_{sk} encryption. Since the adversary $\mathscr{A}$ doesn't have the encryption key sk , with all but negligible probability, the PCPA-security of ε_{sk} will guarantee that each encrypted document C_i and a real ciphertext C'_i is indistinguishable. Similarly, in the process of index construction, since we make use of a pseudo-random function f in $\textbf{Real}_{\mathscr{A}}(k)$, $\mathscr{A}$ is unable to distinguish between the outputs of f and random strings of same size without knowing the key K_2 . So, with all but negligible probability, $\mathscr{I}'$

is distinguishable from real index. Likewise, since $\mathscr{A}$ will not possess K_1 , the pseudo-randomness of π and f will guarantee that$\mathscr{A}$ cannot tell the differences between T_{w_i} in $\mathbf{Real}_{\mathscr{A}}(k)$ and T'_{w_i} in $\mathbf{Sim}_{\mathscr{A},\mathscr{S}}(k)$. Thus, $\mathscr{A}$ cannot distinguish $\mathbf{Real}_{\mathscr{A}}(k)$ and $\mathbf{Sim}_{\mathscr{A},\mathscr{S}}(k)$.

Then, we say our proposed scheme is adaptively semantic secure.

5 Performance Evaluation

5.1 Performance Comparison

We compare our construction with previous symmetric searchable encryption schemes CM-1[8], CM-2 [8], SSE-2 [23] and Sun's scheme [25] Tab. 1 shows the properties and performance of the above schemes. Because the storage overhead of cipher-texts varies for different document encryption method, if we want to make the comparing process easier, we suppose that all the files in C have the same size. For reducing the communication overheads, we only consider the costs of trapdoors and neglect the size of the returned cipher-texts.

Comparing to the scheme CM-1, our proposed scheme does not need to store a dictionary in the user side which reduces data management cost and storage facility spending for users. Comparing to CM-2, the proposed scheme reduces the storage cost in cloud server and takes only one round for each query. In terms of communication, we note that the demands are $O(n)$ in SSE-2 and $O(1)$ in our construction. Our scheme needs far less communication overhead than SSE-2. Among the schemes, only the indexes of SSE-2 and our scheme hide securely the number of distinct words in the dictionary. As a result, the cloud server can't learn statistic information regarding the number of distinct keywords in a specific document from the encrypted index, which can prevent the server from deducing certain document using the known background information. Thus, regarding privacy, our scheme is better.

Table 1: **Comparison Of Various Schemes**

Properties	CM-1	CM-2	SSE-2	Sun's	our scheme
server storage	O(n)	O(n)+O(m)	O(n)	O(n)	O(n)
communication	O(1)	O(1)	O(n)	O(1)	O(1)
number of rounds	1	2	1	1	1
User storage	O(n)	O(1)	O(1)	O(1)	O(1)
Hide m	No	No	Yes	No	Yes

5.2 Experimental Evaluation

To evaluate the efficiency and practicality, we conduct an experimental evaluation for our proposed scheme. The whole experiments are implemented by using C language on a Linux OS equipped with 2.70GHz Intel(R) Pentium(R) CPU and 4GB RAM. In our experiment, we totally select 10000 real data files from the on-line database [2]. All the document collections and extracted keywords are based on the selected data files.

1. Index construction

   In the encrypted index structure, every entry in EEIM stores a subindex. Its length is equal to the number of documents in the data set. To get a subindex, we need to map the corresponding keyword to an index vector and then encrypt the index vector. The time used for calculating a subindex depends on the number of all documents. As Fig. 3 shows, when the size of dictionary

is constant (m=3500), the time spent to build index increases linearly with the size of document collection. According to the Fig. 3, although the process of index construction in our scheme needs to cost much time, the operation is just one-time and off-line. This will not affect the performance evaluation of a scheme too much.

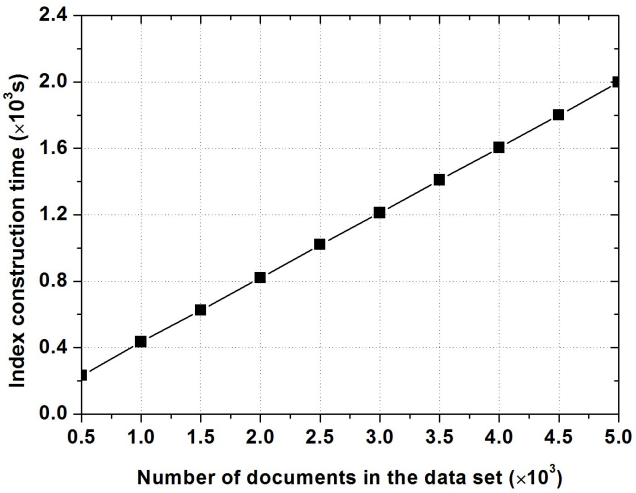


Figure 3: **Time cost to Build index with the constant size of dictionary(m=3500)**

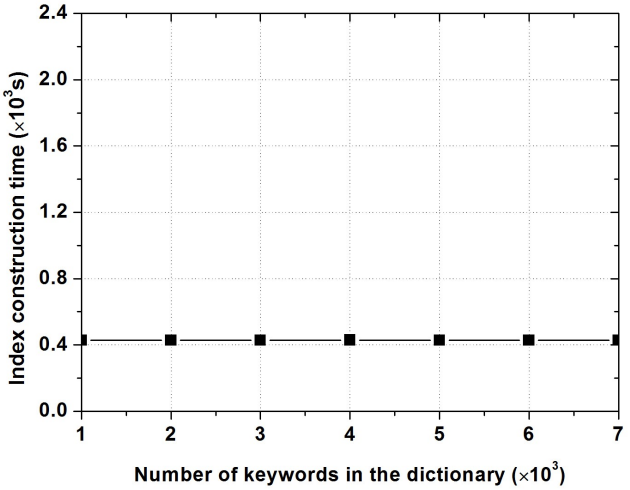


Figure 4: **Time cost to build index with the constant size of document collection (n=1000)**

Since the total number of entries in EEIM is determined by the length of the longest word, the whole index requires $2^d \times n$ bits storage space. Given the same document collection (n=1000), we extract different number of keywords. The Fig. 4 displays that the time cost of building index is almost unchanged. This is because the real size of dictionary is much less than 2^d . From our experimental results, we can infer that the time taken by building index is heavily affected by the number of files in the data set. From the perspective of data users, it is available to set more useful

tokens to keywords of a document, which will not cost additional time.

2. Trapdoor generation

For trapdoor generation, we mainly consider the time to produce pseudo-random bits used to decrypt the corresponding index vector. The length of pseudo-random bits is same with that of index vectors, which is determined by the size of document collection. The evaluated results of generating a trapdoor are shown in Fig. 5. We can observe that the time spent on obtaining a trapdoor is almost linear with the number of documents in the data set. In the test, when the number of files is up to 10000, the trapdoor generation is very fast and just needs less than 0.4 ms.

3. Search

The search operation in our proposed scheme includes locating and decrypting the corresponding entry in EEIM. The time complexity for searching is $O(n)$. As seen from the Fig. 6, the search time grows nearly linearly with the increased number of documents in the data set. We can disclose that when we use 10000 documents to build the index, the time of executing a search algorithm is less than 0.06 ms, which shows our proposed scheme is efficient and quite applicable in the real life.

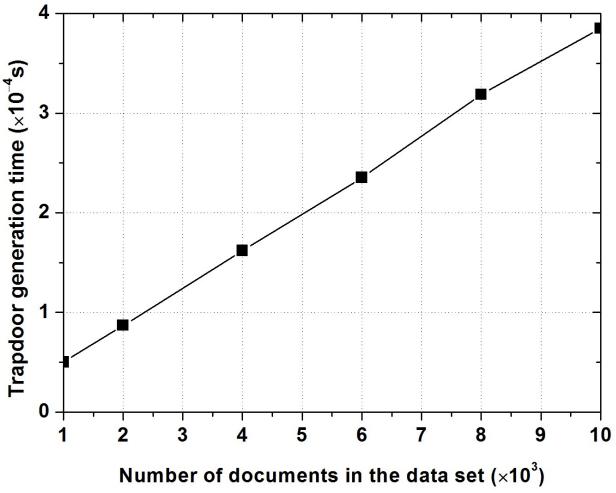


Figure 5: **Time cost to generate trapdoor**

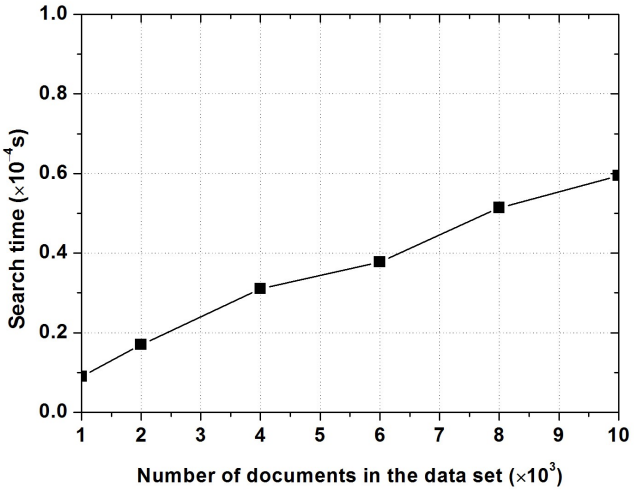


Figure 6: **Time cost to search**

6 Extensions

To enhance the utility of our scheme, here we consider two common situations as extensions to the basic keyword search scheme. One is occurrence queries and the other is dynamic user management.

6.1 Occurrence Queries

In real life, facing large amount of outsourced documents in the cloud, the data user wants to obtain some files which contains more queried keywords. They may send a search request such as "I want to fetch all the files in which 'Panda' appears more than ten times". In order to address the problem of occurrence queries, our scheme can be modified in the following approach.

If a keyword w_i occurs $s_{i,j}$ times in one documentD_j , for every appearance, we label it by concatenating w_i with z, denoted as $w||z$, where z is the order in which w_i appears in the document D_j . We define the set of labels for a keyword w_i in the document D_j as $OCCUR(w_i, j) = \{w_i||z, 1 \leq z \leq s_{i,j}\}$. For example, if the word "Panda" appears three times in D_5 , then we denote $OCCUR(\text{“}Panda\text{”}, 5)$ as $\{\text{“}Panda1\text{”}, \text{“}Panda2\text{”}, \text{“}Panda3\text{”}\}$. In our scheme, only the algorithms BuildIndex$(K, \mathscr{D})$ and Trapdoor(K, w) need to be changed. Improved details are presented as follows.

1. BulidIndex$(K, \mathscr{D})$.

   1) The data owner obtains the distinct keywords from all the documents $\mathscr{D}$ by looking over the whole documents in $\mathscr{D}$. Then he or she generate the dictionary Δ. For each word $w_i \in \delta$ and each document $D_j \in \mathscr{D}$, he/she derives the set of labels for w_i , i.e., $OCCUR(w_i, j) = \{w_i||z, 1 \leq z \leq s_{i,j}\}$, where $s_{i,j}$ is the occurrence number of w_i in the document D_j .

   2) The data owner then builds up an IM-based index structure. Assume that max_i is the maximum value of $s_{i,j} 1 \leq j \leq n$, i.e., $max_i = \max_{1 \leq j \leq n} \{s_{i,j}\}$. For each keyword $w_i \in \delta$, and for , he/she generates a n-bit binary index vector I_{i_z}. For each file D_j,$1 \leq j \leq n$, if $w_i||z \in OCCUR(w_i, j)$, then he/she sets $I_{i_z}[j] = 1$; otherwise, sets $I_{i_z}[j] = 0$.

3) Let $\mathscr{I}$ be an array of size 2^d . For each keyword $w_i \in \delta$ and for $1 \leq z \leq max_i$, the data owner computes $\pi_{K_1}(w_i||z)$ with the random key K_1 and then sets $\mathscr{I}[\pi_{K_1}(w_i||z)] = I_{i_z}$. Other $(2^d - \sum_{i=1}^{m} max_i)$ entries of $\mathscr{I}$ are set to zero vectors with n-bit size.
4) For all $addr = 1, ..., 2^d$, the data owner computes $f_{K_2}(addr)$ with the random key K_2 to obfuscate the original information and sets $\mathscr{I}(addr) = \mathscr{I}(addr) \oplus f_{K_2}(addr)$.
5) The data owner outputs the index $\mathscr{I}$.

2. Trapdoor(K, w).

When the data user wants to search the documents that contain a keyword w at least z times, he/she computes $\pi_{K_1}(w_i||z)$ and $f_{K_2}(\pi_{K_1}(w_i||z))$ using the secret keys K_1 , K_2 , then sends the trapdoor $T_w = (\pi_{K_1}(w_i||z), f_{K_2}(\pi_{K_1}(w_i||z)))$ as a search query to the cloud server.

6.2 Dynamic User Management

Our scheme makes use of the system model of multi-users setting where only the data owner is capable of storing his/her encrypted data on the remote cloud server, whereas an arbitrary user in one defined group is allowed to submit the search queries and read the outsourcing data. This can realize data sharing among a multitude of users while preserving privacy in the untrusted cloud storage environment. A noteworthy problem is how to manage the data users dynamically. In a multi-user environment, when a user is revoked, the user will not have the access privilege to the owner's documents any more.

To solve this problem, we introduce the broadcast encryption (BE) scheme to our scheme. This combination achieves the dynamic management of users. BE is a kind of secure group communication technology, the main characteristic of which is controlling a group of users' access permissions without updating users' keys. In BE, an encrypted message is transported in a broadcasting channel and only the authorized user selected by the message sender can decrypt the message to plain-text. Let S denotes the set of all possible user identifiers, and $G \subseteq S$ the set of users authorized by the data owner. We define a broadcast encryption scheme concluding four polynomial-time algorithms $BE = (Gen, Enc, Add, Dec)$ such that:

1. Gen(1^k) is a probabilistic key generation algorithm. It takes a security parameter k as input and outputs a mask key k_m .

2. Enc(k_m, G, m) is a message encryption algorithm. It takes a mask key k_m, a user collection G and a message m as input, and outputs a cipher-text c .

3. Add(k_m, u) is a user add algorithm. It takes a mask key k_m and a user identifier $u \in G$ as input, and outputs a the user's key k_u.

4. Dec(k_m, c, k_u) is a decryption algorithm. It takes a mask key k_m, an encrypted message c and a user identifier k_u as input, and outputs a plain-text m or a failure feedback.

As stated in our threat model, we can see that the cloud server is "honest-but-curious". We also suppose that the server is deserved to be believed and it will not collude with revoked users to attack the entire system. We now describe the approach in details. Let φ be a pseudo-random permutation: $\varphi : \{0,1\}^k \times \{0,1\}^{d+n} \rightarrow n\{0,1\}^{d+n}$, where $d+n$ is the size of a trapdoor in our scheme. We assume the authorized users include the server. Let k_s denote the server key for BE. The data owner generates the server state sta_s. And sending it to the server. Let sta_G denote the state of the group G, the value of which would be update when a user is removed from G. Next, base on the proposed searchable encryption

scheme, we explain how we manage users' privilege of searching on cloud data from three stages: setup phase, user management phase, and retrieval phase.

In the setup phrase:

1. The data owner chooses a security parameter k and calls $Gen(1^k)$ to generate a mask key k_m of BE. Then he/she calls Keygen(1^k) to generate a secret key $K = (K_1, K_2)$. Let $K_o = (k_m, K)$.

2. The data owner invoke the algorithm BuildIndex$(K, \mathscr{D})$ to construct a secure index $\mathscr{I}$ and encrypt it. The concrete details is same with the work presented in the section III. Then he/she chooses a key $\lambda \in \{0,1\}^k$ randomly and obtains a server state value sta_s by computing Enc(k_m, G, λ). He/She sets the group state value sta_G to be λ. Finally, he/she sends the encrypted documents, $\mathscr{I}$ and sta_s to the cloud server.

In the user management phase:

1. When the data owner wants to share his/her encrypted cloud data with others, he/she can create a group G. In the G, we allow the authorized users to access to the data owner's documents in the cloud server. To add a data user u into G, the data owner computes Add(k_m, u) to get a user key k_u for BE, then sends the triple (K, k_u, λ) to the data user .

2. To remove a data user from the group G, the data owner needs to choose a new key $\lambda' \in 0, 1^k$ and generates a new sta'_s by computing Enc$(k_m, G\backslash u, \lambda')$, where $G\backslash u$ denotes the set excluding the user u . He/She sets sta_G to be λ' , and then sends the new sta'_s to the server. The server will store sta'_s instead of the old value of sta_s .

In the retrieval phase:

1. When a data user u wants to query the keyword w , he/she firstly retrieves the latest server state sta_s from the server, then uses his/her own key k_u to computes Dec(k_u, sta_s). Only if the data user u is still in the group G, will he/she be able to recover the latest key λ that is the output of Dec(k_u, sta_s). Then the data user generates a trapdoor T_w for w with the method introduced in section 3. Finally he/she computes $\varphi_\lambda(T_w)$ using φ keyed with λ and sends the transposition result to the server.

2. Upon receiving the value $\varphi_\lambda(T_w)$, the server gets the key λ by computing Dec(k_u, sta_s). Then, the server computes ${\varphi_\lambda}^{-1}(\varphi_\lambda(T_w))$ to recover the unencrypted trapdoor T_w . At last, the server runs the algorithm $Search(\mathscr{I}, T_w)$ and sends back the encrypted documents which contain the keyword w.

Since the mask key k_m is only known by the data owner. Only the data owner has the permission to update the data set $\mathscr{D}$, add authorized data users and perform user revocation. Once a data user u is revoked, the key λ for the pseudo-random permutation φ will be updated. At this time the user key k_u will become invalid. The revoked user is not able to perform search operations any more. Using this solution, our scheme realizes dynamic management of users.

7 Conclusion

In this paper, we propose an efficient symmetric searchable encryption scheme for cloud storage. The EEIM based search index not only helps users to avoid storing a dictionary in local, but also achieves the search functionality in one interaction. The given security analysis and intensive experiments on the real data set indicates that our proposed scheme realizes privacy preserving and practical search efficiency, which satisfies our design goals. At last, we further present two extended schemes supporting more search scenarios in the cloud computing including occurrence queries and dynamic user management.

Acknowledgement

This research is supported by National Natural Science Foundation of China(61572267), the Open Research Project (2017-MS-21, 2016-MS-23) of State Key Laboratory of Information Security in Institute of Information Engineering, Chinese Academy of Sciences.

References

[1] `http://www.vidhatha.com` [Online; Accessed on May 1, 2017].

[2] `http://www.cs.cmu.edu/enron/` [Online; Accessed on May 1, 2017].

[3] L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *Proc. of the 7th International Conference on Information and Communicatoins Security (ICICS'05), Beijing, China*, volume 3783 of *Lecture Notes in Computer Science*, pages 414–426. Springer, Berlin, Heidelberg, December 2005.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proc. of the 2004 International Conference on the Thoeory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, Berlin, Heidelberg, May 2004.

[5] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proc. of the 4th Conference on Theory of Cryptography (TCC'07), Amsterdam, Netherlands*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Berlin, Heidelberg: Springer-Verlag, February 2007.

[6] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):222–233, January 2014.

[7] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very large databases: Data structures and implementation. In *Proc. of the 21st Annual Network and Distributed System Security Symposium (NDSS'14), San Diego, California, USA*, pages 1–16. Internet Society, February 2014.

[8] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Proc. of the 3rd International Conference on Applied Cryptography and Network Security (ACNS'05), New York, New York, USA*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer, Berlin, Heidelberg, June 2005.

[9] A. P. D. Song, D. Wagner. Practical techniques for searches on encrypted data. In *Proc. of the 2000 IEEE Symposium on Security and Privacy (S&P'00), Berkeley, California, USA*, pages 44–55. IEEE, May 2000.

[10] D.Cash, S.Jarecki, C.Jutla, H.Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Proc. of the 33rd Annual Cryptology Conference: Advances in Cryptology (CRYPTO'13), Santa Barbara, California, USA*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373. Berlin, Heidelberg: Springer-Verlag, August 2013.

[11] Q. Do and F. Hussain. A hybrid approach for the personalisation of cloud-based e-governance services. *International Journal of High Performance Computing and Networking*, 7(3):205–214, September 2013.

[12] E. S. E. Stefanov, C. Papamanthou. Practical dynamic searchable encryption with small leakage. In *Proc. of the 21st Annual Network and Distributed System Security Symposium (NDSS'14), San Diego, California, USA*, pages 1–15. Internet Society, February 2014.

[13] M. Ficco. Security event correlation approach for cloud computing. *International Journal of High Performance Computing and Networking*, 7(3):173–185, September 2013.

[14] Z. Fu, X. Sun, Z. Xia, L. Zhou, and J. Shu. Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing. In *Proc. of the 32nd IEEE International Performance Computing and Communications Conference (IPCCC'14), San Diego, California, USA*, pages 1–8. IEEE, February 2014.

[15] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/` [Online; Accessed on May 1, 2017].

[16] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Proc. of the 2nd International Conference on Applied Cryptography and Network Security (ACNS'04), Yellow Mountaions, China*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer, Berlin, Heidelberg, June 2004.

[17] Y. Hwang and P. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *Proc. of the 1st International Conference on Pairing-based Cryptography (Pairing'07), Tokyo, Japan*, volume 4575 of *Lecture Notes in Computer Science*, pages 2–22. Springer, Berlin, Heidelberg, July 2007.

[18] X. C. J. Li. Efficent multi-user keyword search over encrypted data in cloud computing. *Computing and Informatics*, 32(4):723–738, 2013.

[19] X. Jiang, J. Yu, F. Kong, X. Cheng, and R. Hao. A novel privacy preserving keyword search scheme over encrypted cloud data. In *Proc. of the 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC'15), Krakow, Poland*, pages 836–839. IEEE, November 2015.

[20] P. V. Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In *Proc. of the 7th VLDB Workshop on Secure Data Management (SDM'10), Singapore*, volume 6358 of *Lecture Notes in Computer Science*, pages 87–100. Berlin, Heidelberg: Springer-Verlag, September 2010.

[21] S. K. M. Chase. Structured encryption and controlled disclosure. In *Proc. of the 16th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'10), Singapore*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. Springer,Berlin,Heidelberg, December 2010.

[22] S. Open. Journal of cloud computing. `http://www.journalofcloudcomputing.com` [Online; Accessed on May 1, 2017].

[23] R.Curtmola, J.A.Garay, S.Kamara, and R.Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proc. of the 13th ACM Conference on Copmputer and Communications Security (CSS'06), Alexandria, Virginia, USA*, pages 79–88. ACM, October 2006.

[24] C. P. S. Kamara. Parallel and dynamic searchable symmetric encryption. In *Proc. of the 17th International Conference on Financial Cryptography and Data security (FC'13), Okinawa, Japan*, volume 7859 of *Lecture Notes in Computer Science*, pages 258–274. Berlin, Heidelberg: Springer-Verlag, April 2013.

[25] W. Sun, X. Liu, W. Lou, Y. Hou, and H. Li. Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. In *Proc. of the 12th IEEE Conference on Computer Communications (INFOCOM'15), Hong Kong*, pages 2110–2118. IEEE, May 2015.

[26] A. Swaminathan, Y. Mao, G. Su, H. Gou, A. Varna, S. He, M. Wu, and D. Oard. Confidentiality-preserving rank-ordered search. In *Proc. of the 2007 ACM workshop on Storage Security of Ad hoc and Sensor Networks (SASN'03), Fairfax, Virginia, USA*, pages 7–12. ACM, October 2007.

[27] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. In *Proc. of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS'10), Genova, Italy*, pages 253–262. IEEE, June 2010.

[28] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1467–1479, August 2012.

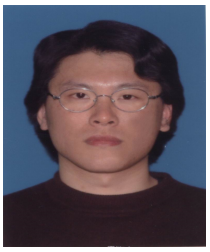
[29] K. Zaerens. Gaining the profits of cloud computing in a public authority environment. *International Journal of Computational Science and Engineering*, 7(4):269–279, October 2012.

[30] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski. Zerber^{+R}: Top-k retrieval from a confidential index. In *Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'09), Saint Peterburg, Russia*, pages 439–449. ACM, March 2009.

Author Biography

Xiuxiu Jiang received B.S. degrees in School of Computer Science and Technology from Qingdao University, China, in 2013. She will receive M.S. degree in the college of Computer Science and Technology from Qingdao University, China, in 2016. Her research is cloud computing security.

Xinrui Ge received B.S. degrees in School of Computer Science and Technology from Qingdao University, China, in 2016. She is currently a graduate student in the College of Computer Science and Technology at Qingdao University, China. Her research is cloud computing security.



Jia Yu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. He is currently an associate professor in the College of Computer Science and Technology at Qingdao University, China. His research interests include key evolving cryptography, digital signature, cryptographic protocol, and network security.

Fanyu Kong received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. He is currently an associate professor in the Institute of Network Security at Shandong University, China. His research interests include cryptanalysis, digital signature, and network security.

Xiangguo Cheng received the B.S. degree in Mathematics Science from Jilin University in 1992, the M.S. degree in Applied Mathematics Science from Tongji University in 1998, and the Ph.D. degree in State Key Laboratory of Integrated Services Network of Xidian University in 2006. He is currently an associate professor in the College of Computer Science and Technology at Qingdao University. His research interests include computer security, public key cryptosystems, and their applications.

Rong Hao received M.S. degree in Institute of Network Security from Shandong University, China, in 2006. She is currently a lecture in the College of Computer Science and Technology at Qingdao University, China. Her research interests include digital signature and secret sharing.