A Secure and Dynamic Fuzzy Multi-Keyword Ranked Search over Encrypted Cloud Data Without Access Pattern Leakage

An-ni CAO

School of Mathematics and Systems Science, Beihang University, Beijing, 100191, China

Keywords: Searchable Encryption, Multi-keyword search, Fuzzy encryption, Access pattern.

Abstract. Recent years, Searchable Encryption (SE) has become one of the hottest topics in cloud computing, because of its flexibility and availability. Most of existing SE schemes focus on the improvement of its efficiency, accuracy, and availability. However, the majority of them leak data access patterns due to their schemes constructions. In this paper, we present a secure symmetric searchable encryption scheme without access pattern leakage, based on Fu’s scheme. By constructing two cloud servers to replace the original server, our scheme protects data users from access pattern leakage.

Introduction

With the rapid development of cloud computing, more and more data owners outsource their data to public cloud server. The main advantages of data outsourcing are that it can save users physical storage space and make the best of cloud server computation capability. However, the privacy of these information is concerned for every data owner, because many sensitive data (e.g. personal emails, bank account information, health care records etc.) are stored in the cloud. The intuitive method used to protect the privacy of outsourced data is encryption these data before outsourcing. It prevents the server or any other adversaries from getting data directly.

But, the advantages of outsourcing storage are lost, if the data owner can’t retrieve documents selectively. Therefore, we need to construct secure and efficient search schemes which make it possible to selectively retrieve outsourced documents. Searchable Encryption (SE) is presented in purpose of solve this problem.

SE scheme enables users to perform keyword search on encrypted outsourced data. In 2000, Song et al. [1] present the idea of searchable encryption firstly. The research of searchable encryption roughly divided into two categories: Symmetric Searchable Encryption (SSE) [2] and Public Key Encryption with Keyword Search (PEKS). Although the PEKS schemes have more functionality, the computation efficiency is faster for SSE schemes compared with PEKS schemes. Therefore, we discuss the Symmetric Searchable Encryption in this article.

The efficiency of algorithm is a hot topic in the research in searchable encryption. The algorithm in [1] is linear to the length of documents. In 2004, Goh et al. [3] presented a new data structure that data owner could build an “index” for the documents set. The efficiency of keyword search is improved by the “index”. However, this algorithm had a high possibility of false positives. With a lot of following works in searchable encryption, there are many schemes are proposed, such as dynamic SSE schemes [4,5], multi-keyword SSE schemes [6,7], etc. In order to support misspelling keyword query search, Li et al. [8] introduced the notion of fuzzy keyword search.

Otherwise, the privacy of SSE schemes is also a main topic to research. For example, if an adversary monitors the communication between users and the server, the adversary can see which keyword that is searched from the query. What’s more, adversaries can estimate whether the same search was performed in the past or not, which is called the leakage of search pattern [9]. But this problem can be solved by randomly generating queries for the same keyword. However, for the inner adversary, it is useless to generate queries randomly, because the search results are fixed for the same keyword. In this way, the adversary gets some information about users privacy data. That is called the leakage of access pattern [10].

In this paper, we present a new symmetric searchable encryption scheme protecting the users
from the leakage of access pattern based on the scheme of Fu et al. [11]. By dividing the function of the original server to two parts, we make it impossible for the two new servers that they can’t estimate the corresponding relationship between the queried keywords and the search results. What’s more, our scheme also has the properties of Fu’s scheme, such as supporting updating, supporting some kinds of misspelling, ranked results according to the relevance score.

**Preliminaries**

**System Model**

As most of SSE schemes, there are three kinds of entities in the cloud service system, listed as the data owner, the data user and the cloud servers. In our system model, data owner has a collection of documents $D = \{D_1, D_2, ..., D_n\}$ and a corresponding table $T$ about the documents and its identifiers. Both are outsourced to the server2 in encrypted form. To support efficient search on the encrypted documents, data owner builds an index $I$ according to the documents set $D$ and the keywords set $W = \{W_1, W_2, ..., W_m\}$ extracted from $D$. Then the secure index $I$ is outsourced to the server1.

If an authorized user wants to search the subset of documents that contain the given keywords, the user should compute a query $q$ and send the secure query $Q$ to server1. The server1 uses $Q$ to search in index $I$ and returns the corresponding set of the identifiers to the server2. Then server2 search in the corresponding table $T$ and return the set of the encrypted documents to users.

Besides, the search results can be ranked by the cloud server and return the top-$K$ relevant files to the user in our scheme.

**Threat Model**

In our threat model, we assume that data owners and data users are trusted. The two servers are ‘honest but curious’ as most SE schemes [8]. It means that the cloud servers may try to obtain some sensitive information from user’s search requests and the retrieval process even though documents have been encrypted. Our security goal is proposing a secure search scheme that allows documents to be securely retrieved while revealing as little information as possible to the clouds. What’s more, we assume that the two servers in our scheme don’t collude to get users information illegally.

**Access Pattern**

To protect the security of searchable encryption scheme, data users prefer to avoid their queries from being exposed to others. For example, if the relationship between keywords and queries is one to one, given a query of a keyword, adversaries can estimate whether the same search was performed in the past or not. This kind of leakage in SE is called the search pattern. Our scheme uses randomly generated queries to protect users from search pattern.

By randomizing token generation algorithm, data owners can defend the outside adversaries of the cloud server. However, how about the inner adversaries in the server (e.g. cloud administrators)? The entry between the index and every search process can disclose the information of keywords, because it always returns the same documents for the same queried keyword whatever the query is. This kind of leakage in SE is called the access pattern. There are some approaches to reduce the access patterns, such as reorder the keywords and regenerate the index periodically.

In our scheme, we use randomly generated queries to protect users from search pattern. What’s more, we propose a new approach to guarantee the privacy of a user’s access pattern.

**Fu’s Scheme**

In our scheme, we use the structure of Fu’s scheme [11]. In this section, we describe the main steps of Fu’s scheme. You can see additional details of this scheme in [11].

**Data Processing.** For a documents set $D = \{D_1, D_2, ..., D_n\}$, data owners extract keywords from $D$ to build a keywords set $W = \{W_1, W_2, ..., W_m\}$ with the Porter Stemming Algorithm [12]. The stemming algorithm aims to find the root of the word. Then data owners compute the relevance between the documents and keywords.
Keyword Transformation. Data owners use a 160-bit unigram vector to represent a keyword (or a query). The detail construction methods of unigram vector in [11]. With the representation, a keyword that might be misspelled in many ways (e.g. misspelling of a letter, missing a letter or adding a letter) can still be represented to a vector that is highly close to the correct one, measured by the Euclidean distance.

Construction of the Bloom-Filter-Based Index/Query. In this step, data owners build the search index with Bloom Filter and Locality-Sensitive Hashing algorithm. Bloom Filter (BF) is a kind of data structure which have high space efficiency. Locality-Sensitive Hashing (LSH) is a kind of algorithm which aim to solve the approximate or exact Near Neighbor Search in high dimensional spaces. LSH hashes ‘close’ points to the same buckets with high probability.

In our scheme, data owners (or data users) use LSH functions to hash the keywords vector (or query vector). For each keyword, it is inserted into a BF by a hash functions family. In this way, every document is represented by a BF, and the index is constructed by all the BFs. According to the properties of LSH functions, two similar inputs within a certain distance are mapped to the same output with high probability. A misspelled keyword is more likely to be hashed into the same bucket in the BF. Finally, the fuzzy keyword search can be achieved.

Inner Product Based Matching Algorithm. The secure index for each document is a Bloom Filter constructed by using the secure kNN algorithm [13]. And every BF contains all keywords in the document. The cloud server can compute the relevance of a given query and every document by doing inner product. If a document contains the query keywords, the inner product result will be a high value. The more queried keywords are contained in the document, the higher the inner product will be. Finally, the top k documents will be returned to data users.

Our Scheme

In this section, we present the formal definition of our scheme. Then describe the construction of the scheme in detail.

Our scheme consists four polynomial-time algorithms:

\[ \Omega = (\text{Setup}, \text{GenIndex}, \text{GenQuery}, \text{Search}), \]

such that:

1. \( \text{SK} \leftarrow \text{Setup}() \)

The data owner generates the secret key set SK, including three parts: 1) a randomly generated \( m \)-bit vector \( S \), 2) two \( m \times m \) invertible matrices \( M_1 \) and \( M_2 \), and 3) a secret key \( K \) used to encrypt documents set D. Namely, \( \text{SK} = \{S, M_1, M_2, K\} \).

2. \( \text{I}, \text{T} \leftarrow \text{GenIndex}() \)

First the unencrypted index is constructed on D by the methods in Fu’s scheme described in the previous section, especially the 1), 2), and 3) steps. What’s more, to solve the access pattern, we generate an \( m \)-bit vectors, called fixed vector, which each bit is 1, and insert it in the index. Secondly, the data owner encrypts the index by the kNN algorithm by the methods in Fu’s scheme. The vectors in index correspond to the file identifiers of documents, instead of the documents.

In our scheme, the data owners construct a one-one map index for all the identifiers and its corresponding documents. The map index and all of the documents can be encrypted by some common methods, such as AES.

Finally, the encrypted index \( \text{I} = \{M_1 \cdot I', M_2 \cdot I''\} \) is sent to cloud1. The encrypted map index T and the encrypted documents set are sent to cloud2.

3. \( \text{Q} \leftarrow \text{GenQuery}() \)

If authenticated data users want to search some keywords in documents set D, the data user first generate a \( m \)-bit-long Bloom Filter as the query. Secondly, the data user randomly generates \( t-1 \) queries as the fake queries, so that the real query can be protected from the leakage of access pattern. Then all queries are encrypted by the kNN algorithm like Fu’s scheme. We assume the real query appears in \( j \)-th position in queries set. And we generate a \( t \)-bit vector, called identification vector, which \( j \) bits randomly is 1. Namely, the query set \( \text{Q} = \{q_1, q_2, ..., q_j, ..., q_{t-1}\} \) and the is sent to cloud1.

4. \( \text{D}' \leftarrow \text{Search}() \)
If the cloud1 received the search queries set Q, he will do search process according Fu’s scheme, and then return the identifiers corresponding to the top-K results. Then cloud1 send the identifiers to cloud2 for further search.

At the same time, the data user establishes a session key with cloud2 by Diffie-Hellman Key Exchange Algorithm. Then once the cloud2 receive the search results of cloud1, he will search his own map index T to retrieve the corresponding documents. Finally, cloud2 uses the session key to encrypt the search results and send to data owner.

Analysis

In this section, we analysis the availability and privacy for our scheme.

1. As the same as Fu’s scheme, our scheme also has the properties the original scheme had. For example:
   - Supporting spelling mistakes: Our scheme should support spelling mistakes, such as misspelling a letter, missing a letter or adding a letter, etc.
   - No Predefined Dictionary: In our scheme, the dictionary is extracted from documents.
   - Supporting updating: The same as original scheme, our scheme support dataset updating, such as file adding, and file deleting.
   - Ranked results: According to relevance score, the server return top-K search results, and the results are ranked.

2. In our scheme, the cloud servers are prevented from obtaining any additional information from the encrypted documents and the index. Because one of the cloud server knows the corresponding relationship between the encrypted query and the identifiers, while the other server knows the corresponding relationship between the identifiers and the encrypted documents subset. If the two servers don’t collude, both of them will know nothing about the access pattern and other information about users.

Summary

In this paper, we focus on the access pattern leakage in symmetric searchable encryption. In 2016, Fu et al. proposed a multi-keyword fuzzy ranked search scheme. Based on Fu’s scheme, we present a secure searchable encryption scheme without access pattern leakage. By separating the original cloud server to two servers who don’t collude, we make both servers get no information about the relationship between keywords and documents. In this way, we protect the users from leakage of access pattern.

References


