Efficient Multi-keyword Ranked Search over Outsourced Cloud Data based on Homomorphic Encryption

Mengxi Nie[1,2], Peng Ran[1] and HaoMiao Yang[1,2]

[1]*University of Electronic Science and Technology of China, 611731 Chengdu, China*
[2]*Science and Technology on Communication Security Laboratory, 610041 Chengdu, China*

Abstract. With the development of cloud computing, more and more data owners are motivated to outsource their data to the cloud server for great flexibility and less saving expenditure. Because the security of outsourced data must be guaranteed, some encryption methods should be used which obsoletes traditional data utilization based on plaintext, e.g. keyword search. To solve the search of encrypted data, some schemes were proposed to solve the search of encrypted data, e.g. top-k single or multiple keywords retrieval. However, the efficiency of these proposed schemes is not high enough to be impractical in the cloud computing. In this paper, we propose a new scheme based on homomorphic encryption to solve this challenging problem of privacy-preserving efficient multi-keyword ranked search over outsourced cloud data. In our scheme, the inner product is adopted to measure the relevance scores and the technique of relevance feedback is used to reflect the search preference of the data users. Security analysis shows that the proposed scheme can meet strict privacy requirements for such a secure cloud data utilization system. Performance evaluation demonstrates that the proposed scheme can achieve low overhead on both computation and communication.

1 Introduction

Cloud computing, a revolution of information technology, can significantly change people's life in the foreseeable future. However, the problem of cloud security hasn't been solved well in practical applications. As we know, cloud computing is based on demand. So how to store data in the cloud is non-transparent for most of data users, which will cause users to worry about the data stored in the cloud be misused or leaked. Though some traditional methods can be used to guarantee the security of cloud, such as authentication, authorization, access control and so on, all of these solutions are based on trusted cloud service providers.

Unfortunately, cloud service providers are not as honest as we wish. A prism door event in 2013 was a typical case that the providers' activities threatened the security of the cloud. According to the report, nine IT giants including Google, Microsoft and Apple open their servers to the United State Security Agency, which makes monitoring worldwide e-mails, calls, and stored dates easily. Because the data are stored on the invisible "cloud" supplied by IT giants, it will undoubtedly give the national security and social stability of one's country an enormous threat.

To ensure privacy, users usually encrypt the data before outsourcing it onto the cloud. However, data encryption makes effective data utilization a very challenging task. It restricts user's ability to search the keywords and fails to satisfy further demands of keyword privacy. Due to these reasons, searchable encryption schemes appear.

Although a series of traditional searchable encryption schemes have been proposed to enable search on ciphertexts, there are still many problems remaining to be solved, e.g. low efficiency, high communication and computing expenditure. In order to address these issues, Yu et al. [1] recently proposed a two-round searchable encryption (TRSE) scheme using vector space model and homomorphic encryption. However, in this scheme, data owners have to encrypt each dimension of the searchable vector which lowers the efficiency. For example, for a vector of dimension n, the data owner will take n encryption operations and output n ciphertexts. In addition, in TRSE, the query vector is generated where each dimension is equal to 0 or 1, which doesn't relate to the search weight of a term.

To achieve low overhead, we propose an efficient multi-keyword ranked searchable scheme (EMRS) over outsourced cloud data based on homomorphic encryption that can pack a vector into only one ciphertext, and thus alleviate the burden of computation and communication. Our contributions can be summarized as follows:

- We adopt a new homomorphic encryption scheme which supports the packing method. Therefore, we can encrypt a vector as a whole in a batch manner. As a result, for a vector of dimension n, we only take one encryption

operation, and thus reduce the computation and communication expenditure.

- We put the search weight of a term into the query vector, i.e., we use values instead of 0 or 1 to fill the query vector. Using this method, the query vector can reflect the data user's search preference.

The rest of this paper is organized as follows: We give the scenario and related background in Section 2. In Section 3, we give the main techniques used in our scheme. In Section4, we present the detailed description of the proposed searchable encryption scheme. Section 5 gives the security analysis of our scheme. Related performance evaluations for both searchable encryption and secure result ranking are discussed in Section 6. Finally, Section 7 gives the concluding remark of the paper.

2 Problem statements

2.1 Scenario

An encrypted cloud data hosting service involves three different entities as illustrated in Fig. 1: the data owner, the data user, and the cloud server.

The cloud server takes the responsibility for holding the third-party data storage and retrieve services. Since there will be some sensitive information in the data and the server cannot be fully entrusted in protecting data, the data must be encrypted before outsourcing. And the leakage of information is unacceptable.

The data owners are group of people who have a collection of n files $C = (f_1, f_2, \cdots, f_n)$ to outsource onto the cloud in encrypted form while still keeping the capability to search through them for effective data utilization reasons. To achieve this, the data owner has to build a searchable index I from a set of l distinct keywords $W = (w_1, w_2, \cdots, w_l)$ extracted from C, and then outsource the index I in encrypted form I' and encrypted files onto the cloud server.

The authorized data user processes multi-keyword retrieval over the outsourced data. Firstly, the data user should generate a query REQ. Secondly, the user encrypts the query and sends it to the cloud server. Finally, the data user gets the relevant files from the cloud server, decrypts it, and then makes use of these files. Note that because of the limitation of user's computing power, operations on the user side should be simplified.

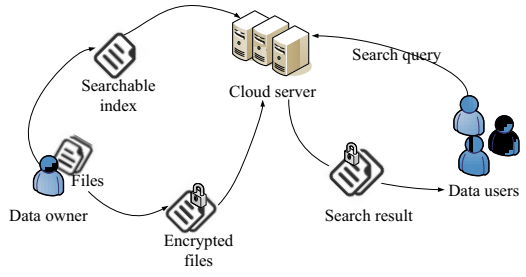

Figure 1. Scenario of retrieval of encrypted cloud data.

2.2 Design Goals

As analyzed above, to realize our efficient and privacy-preserving multi-keyword ranked search over outsourced encrypted cloud data, the following goals should be achieved:

- Confidentiality: The cloud server should not learn plaintexts of data files, index, and searched keywords.

- Unlinkability of trapdoors: The cloud server should not learn whether two searches are performed for the same keywords.

- Preference search: In multi-keyword ranked search, multiple keywords that a data user input are treated as the different weights to reflect the data user' preference.

- Low cost of computation and communication.

3 Preliminaries

3.1 Inner Product Similarity

In our scheme, we use the vector space model to score a file on multi-keywords. A file can be represented as vector $\vec{v_f}$ and each dimension of the vector stands for a separate term, i.e., if this file contains this term, the corresponding value is nonzero, otherwise is zero. Meanwhile, a query can be represented as a vector $\vec{q}$ and each dimension of the vector is padded based on weight. Then we have $score_{f,q} = \vec{v_f} \cdot \vec{q}$. As a result, files can be ranked in order by the scores and the most relevant files can be picked out.

3.2 Relevance Scoring

Here, we adopt the most widely used tf-idf weighting, which involves two attributes: Term Frequency and Inverse document frequency. Term frequency is simply the number of times a given term or keyword appears within a file, and Inverse document frequency is obtained by dividing the number of files in the whole collection by the number of files containing the term.

3.3 Relevance Feedback

Using a Boolean vector to create a query is very imprecise. We use the technique of relevance feedback to add some features to the search query to reflect the search preference of data user [2]-[4].

3.4 BGV Homomorphic Encryption

In this section, we describe a scheme proposed by Brakerski, Gentry, and Vaikuntanathan (BGV) [5], which is regarded as a stripped down vision of FHE scheme and can meet our demand. In our scheme, we use one version of the BGV scheme which deals with integer polynomials and the security of this cryptosystem is linked with the hardness of decisional R-LWE (Ring-Learning with Errors) problem [6]. The BGV scheme is described as follows:

(1)BGV.ParamGen(λ).

The message space is the ring $R_p := \mathbb{Z}_p[X]/\Phi_m(X)$, where $\Phi_m(X)$ is the mth cyclotomic polynomial in $\mathbb{Z}_p[X]$, of degree $n = \varphi(m)$. The ciphertext space is described as follows: pick a large q, which is co-prime to p, and define $R_p := \mathbb{Z}_p[X]/\Phi_m(X)$. The algorithm defines the following distributions:

$D_{\mathbb{Z}^n,\sigma}$: The discrete Gaussian with parameter σ: it is the random variable over $\mathbb{Z}^n$ obtained from sampling $x \in \mathbb{Z}^n$ with probability $e^{-\pi\|x\|/\sigma^2}$ and then rounding at the nearest lattice point.

ZO_n : Sample $x = (x_1, \dots, x_n)$ with $x_i \in \{-1, 0, +1\}$

and Pr[xi = −1] = 1/4; Pr[xi = 1] = 1/4; Pr[xi = 0] = 1/2.

In the following, we assume that the parameters generated here are inputs of any subsequent algorithm.

(2)BGV.KeyGen( )→(*pk*,*dk*).

We obtain a public key encryption scheme, so we need the ability to generate the secret key and the public key. Sample $a \leftarrow R_q$, and s,e $\leftarrow D_{\mathbb{Z}^n,\sigma}$. Let $b \leftarrow as + pe$. We set dk$\leftarrow s$ and $pk \leftarrow (a, b)$.

(3)BGV.Enc$_{pk}$(*m*)→*c*.

Given $m \in R_p$, and sample $u \leftarrow ZO_n$, and $v, w \leftarrow D_{\mathbb{Z}^n,\sigma}$. The output is $c = (c_0, c_1)$, where $c_0 \leftarrow bv + pw + m$ and $c_1 \leftarrow av + pu$.

(4)BGV.Dec$_{dk}$(*c*)→*m*.

Compute $t \in R_q$ as $t \leftarrow c_0 - c_1 s$. The output is then $m = t \bmod p$.

3.5 Smart-Vercauteren ciphertext packing (batch) techniques

One appealing optimization that applies to the protocol in this paper is to use "batch homomorphic encryption" where a single ciphertext represents a vector of encrypted values and a single homomorphic operation on two such ciphertexts applies the homomorphic operation componentwise to the entire vector. In this way, at the cost of a single homomorphic operation, we get to compute on an entire vector of encrypted plaintexts. This is a cryptographic analogue of the Single Instruction Multiple Data (SIMD) architecture and is supported by recent fully homomorphic encryption systems [7].

4 Scheme Design

4.1 Secure Computation of Inner Product

Using homomorphic encryption based on RLWE in our encryption model makes it efficient and secure to calculate the inner production of two vectors. The basic processes are described as follows:

(1) The data user uses the techniques of homomorphic encryption and packing to encrypt all coordinates into a ciphertext. The plaintext space $R_2 = \mathbb{Z}_2[x]/(x^n + 1)$ can be decomposed into *l* plaintext slots: $\mathbb{Z}_2[x]/(f_1(x))$,..., $\mathbb{Z}_2[x]/(f_l(x))$, $x^n + 1 = f_1(x) \cdots f_l(x) \bmod 2$, and each $f_i(x)$, $i = 1, \dots, l$,has the same degree *d*=*n*/*l* only if the system could choose the parameters *n* and *l* properly. Suppose that the data user *A* has a vector $V = (V_1, \dots, V_l) \in \mathbb{Z}^l$ and each dimension of *V* can be encoded into $V_i(x)$, $i = 1, \dots, l$. Then $V_i(x)$, $i = 1, \dots, l$, will be packed into $m_v(x)$ through the Chinese Remainder Theorem (CRT), and an encrypted operation is invoked:$BGV.Enc_{pk}(m_v(x)) \rightarrow c_v(x)$. Similarly, the data user *B* can encrypt another vector *W* into $c_w(x)$.

(2) Through calling one operation of homomorphic multiplication $c(x) = c_v(x) \cdot c_w(x)$, the service provider can compute $C_i(x) = V_i(x) \cdot W_i(x)$, i=1,…,l, parallelly, which is similar to using SIMD.

(3)We call Frobenius isomorphism mapping for *l*-1 times and *l* times of homomorphic addition to deal with *c*(*x*). In details, by using the mapping ϕ to $c^{(i)}(x)$:

$$\phi: x \rightarrow x^2$$

$c^{(0)}(x)$ becomes $c^{(i+1)}(x)$, *i*=0,…,*l*-2, and the corresp-onding plaintext is cyclic shifted for one shot. Then, through calling the operation of homomorphic addition, we can get $c_{inner} = \sum_{i=1}^{l-1} c^{(i)}(x)$, where $c^{(0)}(x)$ is the original $c(x)$.

(4) Finally, by calling $BGV.Dec_{dk}(c_{inner})$ for decrypting, we can get the plaintext of the inner product $m(x) = \sum_{i=1}^{l} V_i(x) \cdot W_i(x)$.

4.2 Relevance Feedback

In our scheme, we use the technique of relevance feedback [2]-[4] instead of a binary vector to create a query aiming at improving the accuracy. Relevance feedback is a process of automatically adjusting an existing query using information feedback by the search entity about the preference of previously retrieved documents. By using the relevance feedback technique, we can optimize the query vector by replacing each dimension with the search weight of a term without change any phase of previous scheme. One of the methods is Rocchio [8] and the main idea is abstracted as follows:

$$Q_{opt} = Q + \beta\left(\frac{1}{N_R}\sum_{j \in U_R} P_j\right) - \gamma\frac{1}{N_N}\sum_{j \in U_N} P_j,$$

Where N_R is the number of relevant data users, U_R is the returned result set and N_N is non-relevant ones U_N; P_j is feature vector of data user U_j; β and γ are suitable

constants [3][4].The search entity can define its preference weights according to the feature vectors retrieved in advance in method. The optimal query can be generated by putting more weights on the relevant features and less weights on the non-relevant features.

Therefore, the cloud server should not only send the identities of the most matching data users, but also return the feature vectors of them in order to define the preference weight using the historical information.

4.3 Privacy-Preserving and Efficient

The framework of EMRS consists of two phases: *Initialization* and *Retrieval*.

The steps of *Initialization* are described as follows:

(1) The data owner calls ***BGV.keygen(λ)*** to get *pk* and *dk,* then assigns *dk* to the authorized data users.

(2) The data owner extracts the collections of *l* keywords, $W = \{w_1, w_2, \dots, w_l\}$ from *n* files, $C = \{f_1, f_2, \dots, f_n\}$ and compute their TF and IDF values. Then, the data owner builds a (*l*+1)-dimensional vector $v_i = \{id_i, t_{i,1}, \dots, t_{i,j}\}$, where $t_{i,j} = tf\text{-}idf_{i,j}$, for each f_i of *C*. The searchable index $I = \{v_i | 1 \le i \le n\}$.

(3) The owner uses the smart-vercauteren ciphertext packing technique to pack the index *I* and then calls $BGV.Enc_{pk}(I)$ to encrypt index *I* into $I' = \{v_i' | 1 \le i \le n\}$.

(4) The owner encrypts *C* into *C′* with cryptology schemes and outsources *C′* and *I′* to the cloud server.

The *Retrieval* phase includes the following steps:

(1) The data user generates a set of query keywords $REQ = \{w_1', w_2', \dots, w_s'\}$, and then the query vector $T_w = \{m_1, \dots, m_l\}$ is built in which m_i is the weight of a query keyword belongs to *REQ*; Othewise, $m_i = 0, 1 \le j \le l$. Then, the user packs the T_w and encrypts it into a trapdoor T_w' and sends T_w' to the server.

(2) For $v_i'(1 \le i \le n)$ in *I′*, the cloud server evaluates $s_i' = v_i'[1; l] \cdot T_w$ homomorphically on ciphertexts and returns $N' = \{(id_1', s_1'), (id_2', s_2'), \dots, (id_n', s_n')\}$ to the user.

(3) The user decrypts *N′* into $N' = \{(id_1, s_1), \dots, (id_n, s_n)\}$, where $s_i = BGV.Dec_{dk}(s_i')$. Note s_i are the relevance scores. Then the data user chooses and sends top-*k* highest-scoring files' identifies $(i_1, \dots, i_k)$ to the server.

(4) The server returns the encrypted *k* files $(f_{i1}, f_{i2}, \dots, f_{ik})$ to the user.

5 Security Analysis

5.1 Confidentiality

Due to the security strength of the file encryption scheme, the file content is clearly well protected. Thus, we only need to focus on the index and the searched keywords. Because the index on the data owner side and the trapdoor on the data user side are encrypted by homomorphic encryption scheme, the privacy of the index and the trapdoor are reduced to the security of the BGV encryption and the packing technique. As showed in [9], the security of BGV and packing is guaranteed by their algorithms, the cloud server can detect nothing about the original data. Namely, the confidentiality will be guaranteed if the secret key dk can be kept properly. It is impossible that the cloud server gets the index vector and queried vector of original files, then infers the TF or ICF of the data files through the secure index vector, secure queried vector and the similarity values. Hence from the analysis, the multikeyword secure ranking algorithm protects the privacy of index, query, and the relevance scores very well.

Additionally, the BGV encryption builds its security on the RLWE hardness problem, the confidentiality in our scheme has a higher level of security than the schemes based on integer[6], e.g. the scheme proposed by Yu. et al [1].

5.2 Unlinkability of trapdoor

The queried vector is encrypted every time before search. The public key *pk* is randomly generated in every encryption, so the searches are independent, i.e., have no relevance. The cloud server cannot mining association rules of the two same secure queried vectors or infer the frequency of keywords in different secure queried vectors through quantities of queried vectors it has received. Namely, in terms of search pattern, the same keywords in different queries, as well as the different keywords in same queries are encrypted independently, e.g., if the same keyword w is requested in different queries, REQ_1 and REQ_2, then the *w* is encrypted into different ciphertexts by calling $BGV.Enc_{pk}()$. As a result, the unlinkability of trapdoor can be achieved. The cloud server cannot mine association rules of the two same secure queried vectors or infer the frequency of keywords in different secure queried vectors through quantities of queried vectors it has received. As a result, the keywords that the data users search can be protected very well.

5.3 Preference search

In multi-keyword ranked search, multiple keywords that a data user input are treated as the different weights to reflect the data user' preference. The vector is encrypted by the secure BGV encryption scheme and its security can refer to Ref. [6]. As a result, the query vector can reflect the data user's preference and its privacy can be protected very well.

6 Performance Evaluation

In this section, we evaluate the performance of the proposed scheme in terms of computation and communication overheads. We denote an addition in $\boldsymbol{G_A}$, a multiplication in $\boldsymbol{G_M}$, an encryption in ***Enc***, and an

isomorphic mapping in G_I. Here, the other operations are neglected, as the cost is very small. Meanwhile, we suppose that the data owner extracts the collections of l keywords.

6.1 Time for encrypting the index

In *Initialization* Phase, the data owner needs to encrypt the searchable index I into I'. The technique of packing makes our scheme has low computation and communication overheads. As shown in Table 1, by using the packing technique, the data owner is able to encrypt one vector for just one time which increases the efficiency.

Table 1. Time for encrypting the index

	TRSE	EMRS
Computing Overhead	The cloud server: $(l\text{-}1)\boldsymbol{G_A}+l\boldsymbol{G_M}$ Data User: l ***Enc***	The cloud server: $1\boldsymbol{G_M} + l\boldsymbol{G_A}+ l\ \boldsymbol{G_I}$ Data User: 1***Enc***
Communication Overhead	Number of ciphertexts: l	Number of ciphertexts: 1

6.2 Performance of the Retrieval Phase

The *Retrieval* phase consists of generating trapdoor, calculating inner production and ranking. We compare the efficiency of this phase between our approach and the approach from TRSE [1].The result is shown in Table 2.

Table 2. Performance of the Retrieval Phase

Scheme	Time
TRSE	$(l+1)$ ***Enc***
EMRS	1***Enc***

Due to the small cost of isomorphic mapping, we can neglect it. Through comparison, we can conclude that our scheme has high efficiency and low computing cost. Besides, the number of ciphertexts in our scheme is extremely small which makes the communication overhead very low.

7 Conclusion

In this paper, we propose an efficient and privacy-preserving multi-keyword searchable scheme over outsourced cloud data based on homomorphic encryption. We use "inner product similarity" to quantitatively evaluate such similarity measure. For reducing the burden of communication and computation, we choose the efficient measure of encryption and inner product calculation which can help us packing the message before dealing with them. Moreover, we put the search weight of a term into the query vector which can reflect the relevance more precisely but not only existence.

Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world data set show our proposed schemes introduce low overhead on both computation and communication.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grants U1233108 and U1333127, the International Science and Technology Cooperation and Exchange Program of Sichuan Province, China under Grant 2015HH0040, and China Postdoctoral Science Foundation funded project under Grant 2014M562309.

References

1. J.D Yu,P Lu,Y.M Zhu,G.T Xue and M.L Li ,Toward Secure Mulitikeyword Top-k Retrieval over Encrypted Cloud Data,,I*EEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, VOL.**10**,NO.**4**,7/8 (2013)
2. G.Salton,M.J.McGill,Introduction to modern information retrieval,McGraw-Hill NewYork(1983)
3. W.M.Shaw Jr,Term-relevance computations and prefect retrieval performance,*Information Processing&Management* 31 (**4**) (1995) 491-498
4. C.Buckley,G.Salton,Optimization of relevance feedback weights,in: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrie-val,351-357(1995)
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping.In Goldwasser [**32**], pages 309-325.
6. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT, pages 1–23, (2010)
7. N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. Des. Codes Cryptogra-phy, 71(**1**):57/81(2014) .Early verion at http://eprint.iacr.org/2011/133.
8. J.J.Rocchio,Relevance feedback in information retrieval, Prentice-Hall,Englewood Cliffs NJ(1971).
9. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. Manuscript, to appear in CRYPTO (2011).