International Journal of Security and Its Applications
Vol.8, No.2 (2014), pp.323-332
http://dx.doi.org/10.14257/ijsia.2014.8.2.33

An Efficient and Privacy-Preserving Semantic Multi-Keyword Ranked Search over Encrypted Cloud Data

Li Chen, Xingming Sun, *Zhihua Xia and Qi Liu

Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China
School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China
z_chenli@163.com, sunnudt@163.com, xia_zhihua@163.com, qrankl@163.com

Abstract

As so much advantage of cloud computing, more and more data owners centralize their sensitive data into the cloud. With a mass of data files stored in the cloud server, it is important to provide keyword based search service to data user. However, in order to protect the data privacy, sensitive data is usually encrypted before outsourced to the cloud server, which makes the search technologies on plaintext unusable. In this paper, we propose a semantic multi-keyword ranked search scheme over the encrypted cloud data, which simultaneously meets a set of strict privacy requirements. Firstly, we utilize the "Latent Semantic Analysis" to reveal relationship between terms and documents. The latent semantic analysis takes advantage of implicit higher-order structure in the association of terms with documents ("semantic structure") and adopts a reduced-dimension vector space to represent words and documents. Thus, the relationship between terms is automatically captured. Secondly, our scheme employ secure "k-nearest neighbor (k-NN)" to achieve secure search functionality. The proposed scheme could return not only the exact matching files, but also the files including the terms latent semantically associated to the query keyword. Finally, the experimental result demonstrates that our method is better than the original MRSE scheme.

Keywords**:** *sensitive data; multi-keyword ranked search; latent semantic*

1. Introduction

Due to the rapid expansion of data, the data owners tend to store their data into the cloud to release the burden of data storage and maintenance [1]. However, as the cloud customers and the cloud server are not in the same trusted domain, our outsourced data may be under the exposure to the risk. Thus, before sent to the cloud, the sensitive data needs to be encrypted to protect for data privacy and combat unsolicited accesses. Unfortunately, the traditional plaintext search methods cannot be directly applied to the encrypted cloud data any more. The traditional information retrieval (IR) has already provided multi-keyword ranked search for the data user. In the same way, the cloud server needs provide the data user with the similar function, while protecting data and search privacy. It is meaningful storing it into the cloud server only when data can be easily searched and utilized.

In the literature, searchable encryption techniques [2-4] are able to provide secure search over encrypted data for users. They build a searchable inverted index that stores a list of mapping from keywords to the corresponding set of files which contain this keyword. When data users input a keyword, a trapdoor is generated for this keyword and then submitted to the cloud server. Upon receiving the trapdoor, the cloud server executes comparison between the trapdoor and index, and finally returns the data users all files that

ISSN: 1738-9976 IJSIA

contain this keyword. But, these methods only allow exact single keyword search.

Some researchers study the problem on secure and ranked search over outsourced cloud data. Wang *et al.*, [5] propose a secure ranked keyword search scheme. Their solution combines inverted index with order-preserving symmetric encryption (OPSE). In terms of ranked search, the order of retrieved files is determined by numerical relevance scores, which can be calculated by $TF \times IDF$. The relevance score is encrypted by OPSE to ensure security. It enhances system usability and saves communication overhead. This solution only supports single keyword ranked search. Cao *et al.*, [6] propose a method that adopts similarity measure of "coordinate matching" to capture the relevance of files to the query. They use "inner product similarity" to measure the score of each file. This solution supports exact multi-keyword ranked search. It is practical, and the search is flexible. Sun *et al.*, [7] proposed a MDB-tree based scheme which supports ranked multi-keyword search. This scheme is very efficient, but the higher efficiency will lead to lower precision of the search results in this scheme.

In addition, fuzzy keyword search [8-10] have been developed. These methods employ a spell-check mechanism, such as, search for "wireless" instead of "wireiess", or the data format may not be the same *e.g.*, "data-mining" versus "datamining. Chuah *et al.*, [8] propose a privacy-aware bed-tree method to support fuzzy multi-keyword search. This approach uses edit distance to build fuzzy keyword sets. Bloom filters are constructed for every keyword. Then, it constructs the index tree for all files where each leaf node a hash value of a keyword. Li *et al.*, [9] exploit edit distance to quantify keywords similarity and construct storage-efficient fuzzy keyword sets. Specially, the wildcard-based fuzzy set construction approach is designed to save storage overhead. Wang *et al.*, [10] employ wildcard-based fuzzy set to build a private trie-traverse searching index. In the searching phase, if the edit distance between retrieval keywords and ones from the fuzzy sets is less than a predetermined set value, it is considered similar and returns the corresponding files. These fuzzy search methods support tolerance of minor typos and format inconsistencies, but do not support semantic fuzzy search. Considering the existence of polysemy and synonymy [11], the model that supports multi-keyword ranked search and semantic search is more reasonable.

In this paper, we will solve the problem of multi-keyword latent semantic ranked search over encrypted cloud data and retrieve the most relevant files. We define a new scheme named Latent Semantic Analysis (LSA)-based multi-keyword ranked search which supports multi-keyword latent semantic ranked search. By using LSA, the proposed scheme could return not only the exact matching files, but also the files including the terms latent semantically associated to the query keyword. For example, when the user inputs the keyword "automobile" to search files, the proposed method returns not only the files containing "automobile", but also the files including the term "car" We take a large matrix of term-document association data and construct a semantic space wherein terms and documents are closely associated are placed near one another. To meet the challenge of supporting such multi-keyword semantic without privacy breaches, we propose the idea: the multi-keyword ranked search (MRSE) using "Latent Semantic Analysis".

The reminder of this paper is organized as follows. In Section Ⅱ, we describe the system model, privacy requirements, and notations. Section Ⅲ provides the detailed description of our proposed mechanism. Section Ⅳ presents the experiment and security analysis. Section Ⅴ summarizes the conclusion.

2. Problem Formulation

A. System Model

The system model can be considered as three entities, as depicted in Figure 1: the data owner, the data user and the cloud server.

Data owner has a collection of data documents $D = \{d_1, d_2, ..., d_m\}$ .A set of distinct keywords $W \equiv \{w_1, w_2, ..., w_n\}$ is extracted from the data collection D . The data owner will firstly construct an encrypted searchable index I from the data collection D . All files in D are encrypted and form a new file collection, C .Then, the data owner upload both the encrypted index I and the encrypted data collection C to the cloud server.

Data user provides t keywords for the cloud server. A corresponding trapdoor T_w through search control mechanisms is generated. In this paper, we assume that the authorization between the data owner and the data user is approximately done.

Cloud server received T_w from the authorized user. Then, the cloud server calculates and returns to the corresponding set of encrypted documents. Moreover, to reduce the communication cost, the data user may send an optional number l along with the trapdoor T so that the cloud server only sends back top-l files that are most relevant to the search query.

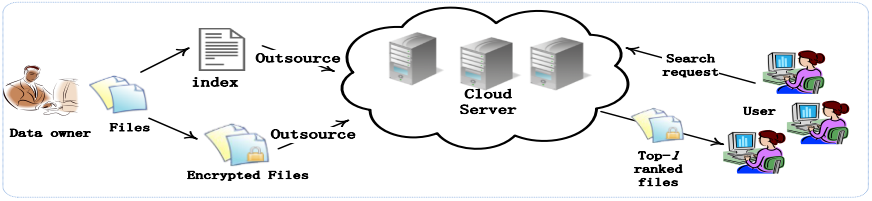

Figure 1. Architecture of Ranked Search over Encrypted Cloud Data

B. Threat models and Design Goals

The cloud server is considered as "honest-but-curious" in our model. Particularly, the cloud server both follows the designated protocol specification but at the same time analyzes data in its storage and message flows received during the protocol so as to learn additional information [12].

In this paper, we purpose to achieve security and ranked search under the above model. The designed goals of our system are following:

Latent Semantic Search: We aim to discover the latent semantic relationship between terms and documents. We use statistical techniques to estimate the latent semantic structure, and get rid of the obscuring "noise" [11].The proposed scheme tries to put similar items near each other in some space in order that it could return the data user the files contain the terms latent semantically associated with the query keyword.

Multi-keyword Ranked Search: It supports both multi-keyword query and support result ranking.

Privacy-Preserving: Our scheme is designed to meet the privacy requirement and prevent the cloud server from learning additional information from index and trapdoor.

1) *Index Confidentiality*. The *TF* values of keywords are stored in the index. Thus, the index stored in the cloud server needs to be encrypted;

2) *Trapdoor Unlinkability*. The cloud server could do some statistical analysis over the search result. Meanwhile, the same query should generate different trapdoors when searched twice. The cloud server should not be able to deduce relationship between trapdoors.

3) *Keyword Privacy.* The cloud server could not discern the keyword in query, index by analyzing the statistical information like term frequency.

C. Notations and Preliminaries

- D --the plaintext document collection, denoted as a set of n data documents $\mathrm{D} = \{d_1, d_2, ..., d_m\}$
- C --the encrypted document collection stored in the cloud server, denoted as $\mathrm{C} = \{c_1, c_2, ..., c_m\}$.
- W --the dictionary, the keyword set composing of m keyword, denoted as $\mathrm{W} \equiv \{w_1, w_2, ..., w_n\}$.
- $\tilde{\mathrm{W}}$ --the subset of W, representing the keywords in a search request.
- I --the searchable index associated C, denoted as $(I_1, I_2, ..., I_m)$.
- $tf_{i,j}$ --the term frequency, the *i-th* term appears times in the *j-th* document.
- $\mathbf{A}'[j]$ --the data vector for document d_j where the element $\mathbf{A}'[i][j]$ represents the term frequency $tf_{i,j}$ of the corresponding keyword W_i in document d_j.
- $\mathbf{Q}$ --the query vector indicating the keywords of interest where each bit $\mathbf{Q}[j] \in \{0,1\}$ represents the existence of the corresponding keyword in the query $\tilde{\mathrm{W}}$.
- $T_{\tilde{w}}$ --the trapdoor for the search request $\tilde{\mathrm{W}}$.
- $\mathrm{D}_{\tilde{w}}$ -the ranked id list of all documents according to their relevance to $\tilde{\mathrm{W}}$.

We now introduce some necessary information retrieval back-group for our proposed scheme:

***Latent Semantic Analysis*:** In information retrieval, latent semantic analysis is a solution for discovering the latent semantic relationship. It adopts singular-value decomposition, which is abbreviated as *SVD* to find the semantic structure between terms and documents. *SVD* can be viewed as a technique for exacting a set of uncorrelated indexing variables or factors. In this paper, the term-document matrix consists of n rows, each of which represents the data vector for each file,

$$\mathbf{A}' = (\mathbf{A}'[1] \quad \cdots \quad \mathbf{A}'[j] \quad \cdots \quad \mathbf{A}'[m]) = \begin{pmatrix} tf_{1,1} & \cdots & tf_{1,j} & \cdots & tf_{1,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ tf_{i,1} & \cdots & tf_{i,j} & \cdots & tf_{i,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ tf_{n,1} & \cdots & tf_{n,j} & \cdots & tf_{n,m} \end{pmatrix} \tag{1}$$

as depicted in the Eq.1. Then, we take a large term-document matrix and decompose it into a set of k, orthogonal factors from which the original matrix can be approximated by linear combination [11]. For example, a term-document matrix named $\mathbf{A}'$ can be decomposed into the product of three other matrices:

$$\underset{n\times m}{\mathbf{A}'} = \underset{n\times t}{\mathbf{U}'} \cdot \underset{t\times t}{\mathbf{S}'} \cdot \underset{t\times m}{\mathbf{V}'} \tag{2}$$

such that $\mathbf{U}'$ and $\mathbf{V}'$ have orthonormal columns, $\mathbf{S}'$ is diagonal. This is so-called singular value decomposition of $\mathbf{A}'$. The singular values in $\mathbf{S}'$ are ordered by size, the first k largest need be kept and the remaining smaller ones set to zero. In short, we choose previous k columns of $\mathbf{S}'$, and then deleting the corresponding columns of $\mathbf{U}'$ and $\mathbf{V}'$ respectively. The product of the resulting matrices is a matrix $\mathbf{A}$ which is only approximately equal to $\mathbf{A}'$, and is the matrix of rank k. The result is a reduced model:

$$\underset{n\times m}{\mathbf{A}} = \underset{n\times k}{\mathbf{U}'} \cdot \underset{k\times k}{\mathbf{S}'} \cdot \underset{k\times m}{\mathbf{V}'} \approx \mathbf{A}' \tag{3}$$

which is the rank-k model with the best possible least-squares-fit to $\mathbf{A}'$ [11, 13].

Secure k-NN: In order to compute the inner product in a privacy-preserving method,

we will adapt the secure k -nearest neighbor scheme[14].In this scheme, distances between database points to a query point are computed for finding the nearer neighbors to the query point. Namely, we could select k nearest records by secure *k-NN* .This splitting technique is secure against known-plaintext attack, which is roughly equal in security to a *d*-bit symmetric key. Here we denote the data vector as $\mathbf{p}$, the query vector as $\mathbf{q}$, and the secret key is composed of one n bit vector as $\mathbf{X}$ and two $n \times n$ inventible matrices as $\{\mathbf{M}_1, \mathbf{M}_2\}$. $\mathbf{q}$ is split into two random vector as $\{\mathbf{q}', \mathbf{q}''\}$. $\mathbf{p}$ is split into two random vector as $\{\mathbf{p}', \mathbf{p}''\}$. Note here the vector $\mathbf{X}$ function as a splitting indicator. If the j -th bit of $\mathbf{X}$ is 0, $\mathbf{p}'[j]$ and $\mathbf{p}''[j]$ are set as the same as $\mathbf{p}[j]$,while $\mathbf{q}'[j]$ and $\mathbf{q}''[j]$ are set to the random numbers so that their sum is equal to $\mathbf{q}[j]$ .If the j -th bit of $\mathbf{X}$ is 1,the splitting process is similar except that $\mathbf{p}$ and $\mathbf{q}$ are switched. The split data vector pair $\{\mathbf{p}', \mathbf{p}''\}$ is encrypted as $\{\mathbf{M}_1^T \cdot \mathbf{p}', \mathbf{M}_2^T \cdot \mathbf{p}''\}$, and the split query vector pair $\{\mathbf{q}', \mathbf{q}''\}$ is encrypted as $\{\mathbf{M}_1^{-1} \cdot \mathbf{q}', \mathbf{M}_2^{-1} \cdot \mathbf{q}''\}$. The formula to calculate the score is,

$$\begin{aligned} &\{\mathbf{M}_1^T \mathbf{p}', \mathbf{M}_2^T \mathbf{p}''\} \cdot \{\mathbf{M}_1^{-1} \mathbf{q}', \mathbf{M}_2^{-1} \mathbf{q}''\} \\ &= \mathbf{M}_1^T \mathbf{p}' \cdot \mathbf{M}_1^{-1} \mathbf{q}' + \mathbf{M}_2^T \mathbf{p}'' \cdot \mathbf{M}_2^{-1} \mathbf{q}'' \\ &= \mathbf{p}'^T \cdot \mathbf{q}' + \mathbf{p}''^T \cdot \mathbf{q}'' \\ &= \mathbf{p}'^T \cdot \mathbf{q}' \end{aligned} \tag{4}$$

Clearly, the score is not affected by encryption .Without prior knowledge of processes, neither query vector nor data vector can be guessed by analyzing their corresponding cipher text. We can get a detailed presentation from the paper [6, 15].

3. Proposed Scheme

In this section, we give a detailed description of our scheme. We firstly propose to employ “Latent Semantic Analysis” to implement the latent semantic multi-keyword ranked search.

A. Our Scheme

Data owner wants to outsource m data files $\mathrm{D} = \{d_1, d_2, \ldots, d_m\}$ that he prepares to outsource to the cloud server in encrypted form while still keeping the capability to search through them. To do so, data owner firstly builds a secure searchable index I from a set of n distinct keywords W extracted from the file collection D . According to the above definition about *LSA*, the data owner builds a term-document matrix $\mathbf{A}'$. Matrix $\mathbf{A}'$ can be decomposed into the product of three other matrices. And then, we reduce the dimensions of the original matrix $\mathbf{A}'$ to get a new matrix $\mathbf{A}$ which is calculated the best “reduced-dimension” approximation to the original term-document matrix [16]. With t keywords of interest in W as input, one binary vector $\mathbf{Q}$ is generated where each bit $\mathbf{Q}[j]$ indicates whether $\mathrm{W}_j \in \mathrm{W}$ is true or false. The similarity score is expressed as the inner product of data vector $\mathbf{A}[j]$ and query vector $\mathbf{Q}$ .Specially, $\mathbf{A}[j]$ denotes the j -*th* column of the matrix $\mathbf{A}$.

- **Setup** The data owner generates a $n+2$ -bit vector as $\mathbf{X}$ and two $(n+2) \times (n+2)$ invertible matrices $\{\mathbf{M}_1, \mathbf{M}_2\}$ .The secret key *SK* is the form of a 3-tuple as $\{\mathbf{X}, \mathbf{M}_1, \mathbf{M}_2\}$.
- **BuildIndex($\mathbf{A}', SK$)** The data owner extracts a term-document matrix $\mathbf{A}'$ from D and decomposes it into three other matrices $\mathbf{U}'_{n \times t}, \mathbf{S}'_{t \times t}, \mathbf{V}'_{t \times m}$. According to our scheme, the data owner adopts statistical techniques to estimate the latent structure, and get rid of the obscuring “noise”. To reduce dimensions, we choose previous k columns of $\mathbf{S}'$, and then deleting the corresponding columns of $\mathbf{U}'$ and $\mathbf{V}'$ respectively. Following, we multiply

these three matrices $\mathbf{U}'_{n\times k}$, $\mathbf{S}'_{k\times k}$, $\mathbf{V}'_{k\times m}$ to get the result matrix $\mathbf{A}$. Taking privacy into consideration, it is necessary that the matrix $\mathbf{A}$ is encrypted before outsourcing. After applying dimension-extending, the original $\mathbf{A}[j]$ is extended to $(n+2)$-dimensions, instead of n. Namely, the $(n+1)$-*th* entry in $\mathbf{A}[j]$ is set to a random number ε_j, and the $(n+2)$-*th* entry in $\mathbf{A}[j]$ is set to 1 during the dimension extending. Finally, $\mathbf{A}[j]$ can be represented as $((\mathbf{A}[j])^T, \varepsilon_j, 1)^T$. The subindex $I_j = \{\mathbf{M}_1^T \cdot \mathbf{A}'[j], \mathbf{M}_2^T \cdot \mathbf{A}''[j]\}$ is built.

- **Trapdoor($\tilde{\mathbb{W}}$)** With t keywords of interest in $\tilde{\mathbb{W}}$ as input, one binary vector $\mathbf{Q}$ is generated. The $(n+1)$-*th* entry in $\mathbf{Q}$ is set to a random number 1, and then scaled by a random number $r \neq 0$, and the $(n+2)$-*th* entry in $\mathbf{Q}$ is set to another random number t during the dimension extending. $\mathbf{Q}$ can be represented as $(r\mathbf{Q}^T, r, t)^T$. After applying the same splitting and encryption as above, the trapdoor $T_{\tilde{w}}$ is generated as $\{\mathbf{M}_1^{-1} \cdot \mathbf{Q}', \mathbf{M}_2^{-1} \cdot \mathbf{Q}''\}$.
- **Query($T_{\tilde{w}}, l, I$)** The inner product of I_j and $T_{\tilde{w}}$ is calculated by the cloud server. After sorting all scores, the cloud server returns the top-l ranked id list $\mathrm{D}_{\tilde{w}}$ to the data user.

The final similarity scores would be:

$$\begin{aligned} I_j \cdot T_{\tilde{w}} &= \{\mathbf{M}_1^T \cdot \mathbf{A}'[j], \mathbf{M}_2^T \cdot \mathbf{A}''[j]\} \cdot \{\mathbf{M}_1^{-1} \cdot \mathbf{Q}', \mathbf{M}_2^{-1} \cdot \mathbf{Q}''\} \\ &= (\mathbf{A}'[j])^T \cdot \mathbf{Q}' + (\mathbf{A}''[j])^T \cdot \mathbf{Q}'' \\ &= (\mathbf{A}[j])^T \cdot \mathbf{Q} \\ &= (\mathbf{A}[j], \varepsilon_j, 1) \cdot (r\mathbf{Q}^T, r, t)^T \\ &= r(\mathbf{A}[j] \cdot \mathbf{Q}^T + \varepsilon_j) + t \end{aligned} \tag{5}$$

Note that in our scheme, we add some random numbers to the final score, which clearly displays security strength.

4. Performance and Security Analysis

In this section, we show a thorough experimental evaluation of the proposed technique on a real dataset: the *MED* dataset [17] .The whole experiment is implemented by C++ language on a computer with Core 2.83GHz Processor, on Windows 7 system. For the proposed scheme, we will reduce to separate dimensions. The performance of our method is compared with the original MRSE scheme.

D. Efficiency

The proposed scheme is depicted in details in previous section, except the KeyGen algorithm. In our scheme, we adopt Gauss-Jordan to compute the inverse matrix. The time of generating key is decided by the scale of the matrix. Besides, the proposed scheme that processed by *SVD* algorithm will consume time. Other algorithms, such as index construction, trapdoor generation, query, which is put forward by us, are consistent with the original MRSE in time-consuming.

E. F-measure

In this paper, we still use the measure of traditional information retrieval. Before the introduction of the F-measure's concept, we will firstly give the brief of the precision and recall. Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance[18]. F-measure that combines precision and recall is the harmonic mean of precision and recall[19]. Here, we adopt F-measure to weigh the result of our experiments.

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{6}$$

F. Performance Analysis

For a clear comparison, our proposed scheme attains score higher than the original *MRSE* in F-measure. Since the original scheme employs exact match, it must miss some similar words which is similar with the keywords. However, our scheme can make up for this disadvantage, and retrieve the most relevant files. Figure 2 shows that our method achieves remarkable result.

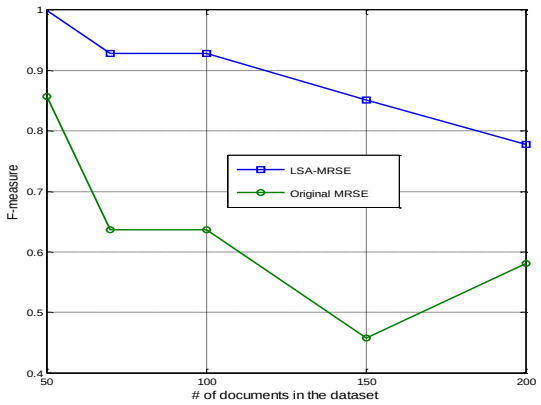

Figure 2. Comparison of Two Schemes

Compared with the traditional vector space, the smaller the latent semantic space is, the more clearly the semantic relationships. Yet, the fact is that the lower dimension will not bring the better result. For example, we will use the 100 documents of *MED* to do the test and reduce separate dimensions respectively. Figure 3 shows a recall-dimension curve. From the Figure 3, the dimension reduces from 100 to 30, the recall has no change. It means that the relevant documents can be retrieved. Obviously, after the dimensions descended to 30, the values of the recall go down .It means that some relevant documents can not be searched. Thus, when we conduct the experiments, we need to choose the appropriate dimension to achieve the best effect of experiment.

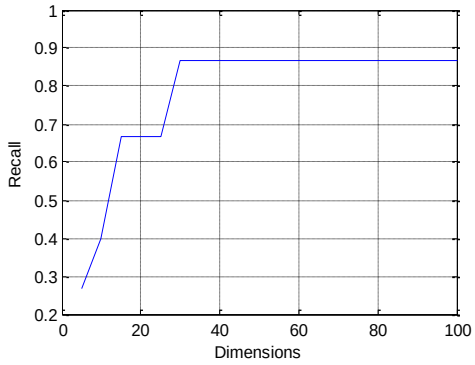

Figure 3. The Recall of Separate Dimensions

G. Security Analysis

We analyze our proposed scheme according to the predefined privacy requirements in design goals:

1) *Index Confidentiality*. In our proposed scheme, I_i and T_w are obfuscated vectors, which means the cloud server can not infer the original data vector and the query vector without the secret key *SK*. As is proven in [14], the cloud server cannot deduce *TF* values from the result relevance scores. In other word, the index confidentiality is protected.

2) *Trapdoor Unlinkability*. The trapdoor of query vector is generated from a random splitting operation, which means the same search requests are transformed into different query trapdoors. And thus, the query unlinkability is well preserved.

3) *Keyword Privacy*. In the known background scheme, the cloud server is supposed to have more knowledge, such as the distribution *TF* values of keywords in the dataset. The cloud server is able to identify keywords by analyzing these specific distributions. In our scheme, the *TF* distributions of keywords will be leaked directly when there is only one query keyword. Thus, our proposed scheme is designed to obscure the *TF* distributions of keywords with the dummy values. That is to say, the keyword privacy is protected.

5. Conclusion

In this paper, a multi-keyword ranked search scheme over encrypted cloud data is proposed, which meanwhile supports latent semantic search. We use the vectors consisting of *TF* values as indexes to documents. These vectors constitute a matrix, from which we analyze the latent semantic association between terms and documents by LSA. Taking security and privacy into consideration, we employ a secure splitting *k-NN* technique to encrypt the index and the queried vector, so that we can obtain the accurate ranked results and protect the confidence of the data well. The proposed scheme could return not only the exact matching files, but also the files including the terms latent semantically associated to the query keyword.

As our future work, we will concentrate on the encrypted data of semantic keyword search in order that we can confront with the more sophisticated search.

Acknowledgments

This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133), GYHY201206033, 201301030, 2013DFG12860, BC2013012 and PAPD fund.

References

[1] M. Armbrust, "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, **(2010)**, pp. 50-58.
[2] D. Boneh, "Public key encryption with keyword search", Advances in Cryptology-Eurocrypt 2004, Springer, **(2004)**.
[3] R. Curtmola, "Searchable symmetric encryption: improved definitions and efficient constructions", Proceedings of the 13th ACM conference on Computer and communications security, ACM, **(2006)**.
[4] D. X. Song, D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data. in Security and Privacy", 2000. S&P 2000, Proceedings 2000 IEEE Symposium, IEEE, **(2000)**.
[5] C. Wang, "Secure ranked keyword search over encrypted cloud data", Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference, IEEE, **(2010)**.
[6] N. Cao, "Privacy-preserving multi-keyword ranked search over encrypted cloud data", INFOCOM, 2011 Proceedings IEEE, IEEE, **(2011)**.

[7] W. Sun, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking", Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, **(2013)**.
[8] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data", Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference, IEEE, **(2011)**.
[9] S. Deshpande, "Fuzzy keyword search over encrypted data in cloud computing", World Journal of Science and Technology, vol. 2, no. 10, **(2013)**.
[10] C. Wang, "Achieving usable and privacy-assured similarity search over outsourced cloud data", INFOCOM, 2012 Proceedings IEEE, IEEE, **(2012)**.
[11] S. C. Deerwester, "Indexing by latent semantic analysis", JASIS, vol. 41, no. 6, **(1990)**, pp. 391-407.
[12] S. Zerr, "Zerber+ r: Top-k retrieval from a confidential index", Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, **(2009)**.
[13] G. W. Furnas, "Information retrieval using a singular value decomposition model of latent semantic structure", Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, **(1988)**.
[14] W. K. Wong, "Secure kNN computation on encrypted databases", Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, ACM, **(2009)**.
[15] C. Yang, "A Fast Privacy-Preserving Multi-keyword Search Scheme on Cloud Data", Cloud and Service Computing (CSC), 2012 International Conference, IEEE, **(2012)**.
[16] S. T. Dumais, "Latent semantic indexing (LSI) and TREC-2", NIST SPECIAL PUBLICATION SP, **(1994)**, pp. 105-105.
[17] MED Dataset. Available from: http://web.eecs.utk.edu/research/lsi/.
[18] D. M. Powers, "The problem with kappa", Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, **(2012)**.
[19] D. L. Olson and D. Delen, "Advanced data mining techniques", [electronic resource] Springer, **(2008)**.

Authors

Li Chen She is currently pursuing her MS in computer science and technology at the College of Computer and Software, in Nanjing University of Information Science and Technology, China. Her research interests include keyword search over encrypted cloud data.

Xingming Sun Prof. Xingming Sun received his BS in mathematics from Hunan Normal University, China, in 1984, MS in computing science from Dalian University of Science and Technology, China, in 1988, and PhD in computing science from Fudan University, China, in 2001. He is currently a professor in School of Computer & Software, Nanjing University of Information Science & Technology, China. His research interests include network and information security, digital watermarking.

Zhihua Xia (Corresponding author) Dr. Zhihua Xia received his BE in Hunan City University, China, in 2006, PhD in computer science and technology from Hunan University, China, in 2011. He works as a lecturer in School of Computer & Software, Nanjing University of Information Science & Technology. His research interests include cloud security, and digital forensic.

Qi Liu Dr. Qi Liu received his BSc degree in Computer Science and Technology from Zhuzhou Institute of Technology, China in 2003, and his MSc and PhD in Data Telecommunications and Networks from the University of Salford, UK in 2006 and 2010. His research interests include context awareness, data communication in MANET and WSN, and smart grid. His recent research work focuses on intelligent agriculture and meteorological observation systems based on WSN.