Dynamic Multi-Keyword Ranking Scheme on Encrypted Cloud Data

K.NARENDRA
PG Student (Software Engineering)
St. Johns College of Engineering and Technology
Kurnool, Andhra Pradesh, India

GKV.NARASIMHAREDDY
Associate professor, Department of CSE
St. Johns College of Engineering and Technology
Kurnool, Andhra Pradesh, India

Abstract—**Due to the increasing popularity of cloud computing, more and more data owners are motivated to outsource their data to cloud servers for great convenience and reduced cost in data management. However, sensitive data should be encrypted before outsourcing for privacy requirements, which obsoletes data utilization like keyword-based document retrieval. In this paper, we present a secure multi-keyword ranked search scheme over encrypted cloud data, which simultaneously supports dynamic update operations like deletion and insertion of documents. Specifically, the vector space model and the widely-used TF_IDF model are combined in the index construction and query generation. We construct a special tree-based index structure and propose a "Greedy Depth-first Search" algorithm to provide efficient multi-keyword ranked search. The secure kNN algorithm is utilized to encrypt the index and query vectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. In order to resist statistical attacks, phantom terms are added to the index vector for blinding search results. Due to the use of our special tree-based index structure, the proposed scheme can achieve sub-linear search time and deal with the deletion and insertion of documents flexibly. Extensive experiments are conducted to demonstrate the efficiency of the proposed scheme.**

Index Terms—**Searchable Encryption, Multi-Keyword Ranked Search, Dynamic Update, Cloud Computing.**

I. INTRODUCTION

CLOUD computing has been considered as a new model of enterprise IT infrastructure, which can organize huge resource of computing, storage and applications, and enable users to enjoy ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources with great efficiency and minimal economic overhead [1]. Attracted by these appealing features, both individuals and enterprises are motivated to outsource their data to the cloud, instead of purchasing software and hardware to manage the data themselves. Despite of the various advantages of cloud services, outsourcing sensitive information (such as e-mails, personal health records, company finance data, government documents, etc.) to remote servers brings privacy concerns.

The cloud service providers (CSPs) that keep the data for users may access users' sensitive information without authorization. A general approach to protect the data confidentiality is to encrypt the data before outsourcing [2]. However, this will cause a huge cost in terms of data usability. For example, the existing techniques on keyword-based information retrieval, which are widely used on the plaintext data, cannot be directly applied on the encrypted data. Downloading all the data from the cloud and decrypt locally is obviously impractical.

This paper proposes a secure tree-based search scheme over the encrypted cloud data, which supports multi keyword ranked search and dynamic operation on the document collection. Specifically, the vector space model and the widely-used "term frequency (TF) × inverse document frequency (IDF)" model are combined in the index construction and query generation to provide multi keyword ranked search. In order to obtain high search efficiency, we construct a tree-based index structure and propose a "Greedy Depth-first Search" algorithm based on this index tree. Due to the special structure of our tree-based index, the proposed search scheme can flexibly achieve sub-linear search time and deal with the deletion and insertion of documents. The secure kNN algorithm is utilized to encrypt the index and query vectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. To resist different attacks in different threat models, we construct two secure search schemes: the basic dynamic multi-keyword ranked search (BDMRS) scheme in the known cipher text model, and the enhanced dynamic multi-keyword ranked search (EDMRS) scheme in the known background model. Our contributions are summarized as follows:

1) We design a searchable encryption scheme that supports both the accurate multi-keyword ranked search and flexible dynamic operation on document collection.

2) Due to the special structure of our tree-based index, the search complexity of the proposed scheme is fundamentally kept to logarithmic. And in practice, the proposed scheme can achieve higher search efficiency by executing our "Greedy

Depth-first Search" algorithm. Moreover, parallel search can be flexibly performed to further reduce the time cost of search process.

II. RELATED WORK

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keyword search over cipher text domain. Due to different cryptography primitives, searchable encryption schemes can be constructed using public key based cryptography [5], [6] or symmetric key based cryptography [7], [8], [9], [10]. Song *et al.* [7] proposed the first symmetric searchable encryption (SSE) scheme, and the search time of their scheme is linear to the size of the data collection. Goh [8] proposed formal security definitions for SSE and designed a scheme based on Bloom filter. The search time of Goh's scheme is $O(n)$, where n is the cardinality of the document collection. Curtmola *et al.* [10] proposed two schemes (SSE-1 and SSE-2) which achieve the optimal search time. Their SSE-1 scheme is secure against chosen-keyword attacks (CKA1) and SSE-2 is secure against adaptive chosen-keyword attacks (CKA2). These early works are single keyword boolean search schemes, which are very simple in terms of functionality. Multi-keyword boolean search allows the users to input multiple query keywords to request suitable documents. Among these works, conjunctive keyword search schemes [15], [16], [17] only return the documents that contain all of the query keywords. Disjunctive keyword search schemes [18], [19] return all of the documents that contain a subset of the query keywords. Predicate search schemes [20], [21], [22] are proposed to support both conjunctive and disjunctive search. All these multi keyword search schemes retrieve search results based on the existence of keywords, which cannot provide acceptable result ranking functionality. Ranked search can enable quick search of the most relevant data. Sending back only the top-k most relevant documents can effectively decrease network traffic. Some early works [23], [24], [25] have realized the ranked search using order-preserving techniques, but they are designed only for single keyword search. Cao *et al.* [26] realized the first privacy-preserving multi-keyword ranked search scheme, in which documents and queries are represented as vectors of dictionary size. With the "coordinate matching", the documents are ranked according to the number of matched query keywords.

However, Cao *et al.*'s scheme does not consider the importance of the different keywords, and thus is not accurate enough. In addition, the search efficiency of the scheme is linear with the cardinality of document collection. Sun *et al.* [27] presented a secure multi-keyword search scheme that supports similarity-based ranking. The authors constructed a searchable index tree based on vector space model and adopted cosine measure together with TF×IDF to provide ranking results. Sun *et al.*'s search algorithm achieves better-than-linear search efficiency but results in precision loss. O¨ rencik *et al.* [28] proposed a secure multi-keyword search method which utilized local sensitive hash (LSH) functions to cluster the similar documents. The LSH algorithm is suitable for similar search but cannot provide exact ranking. In [29], Zhang *et al.* proposed a scheme to deal with secure multi-keyword ranked search in a multi-owner model. In this scheme, different data owners use different secret keys to encrypt their documents and keywords while authorized data users can query without knowing keys of these different data owners. The authors proposed an "Additive Order Preserving Function" to retrieve the most

III. PROBLEM FORMULATION

3.1 Notations and Preliminaries

• W – The dictionary, namely, the set of keywords, denoted as $W = \{w1,w2,..,wm\}$.
• m – The total number of keywords in W.
• Wq – The subset of W, representing the keywords in the query.
• F – The plaintext document collection, denoted as a collection of n documents $F = \{f1, f2,..,fn\}$. Each document f in the collection can be considered as a sequence of keywords.
• n – The total number of documents in F.
• C – The encrypted document collection stored in the cloud server, denoted as $C = \{c1,c2,..,cn\}$.
• T – The unencrypted form of index tree for the whole document collection F.
• I – The searchable encrypted tree index generated from T .
• Q – The query vector for keyword set Wq.
• TD – The encrypted form of Q, which is named as trapdoor for the search request.
• Du – The index vector stored in tree node u whose dimension equals to the cardinality of the dictionary W. Note that the node u can be either a leaf node or an internal node of the tree.
• Iu – The encrypted form of Du

Vector Space Model and Relevance Score Function.

Vector space model along with TF×IDF rule is widely used in plaintext information retrieval, which efficiently supports ranked multi-keyword search [34]. Here, the term frequency (TF) is the number of times a given term (keyword) appears within a document, and the inverse document frequency (IDF) is obtained through dividing the cardinality of document collection by the number of documents containing the keyword. In the vector space model, each document is denoted by a vector, whose elements are the normalized TF values of keywords in this document. Each query is also denoted as a vector Q, whose elements are the normalized IDF values of query keywords in the document collection. Naturally, the lengths of both the TF vector and the IDF vector are equal to the total number of keywords, and the dot product of the TF vector *Du* and the IDF vector *Q* can be calculated to quantify the relevance between the query and corresponding document. Following are the notations used in our relevance evaluation function:

- *Nf;wi* – The number of keyword *wi* in document *f*.
- *N* – The total number of documents.
- *Nwi* – The number of documents that contain keyword *wi*.
- *TF'f;wi* – The TF value of *wi* in document *f*.
- *IDF'wi* – The IDF value of *wi* in document collection.
- *TFu;wi* – The normalized TF value of keyword *wi* stored in index vector *Du*.
- *IDFwi* – The normalized IDF value of keyword *wi* in document collection.

The relevance evaluation function is defined as:

$$\text{RScore}(D_u, Q) = D_u \cdot Q = \sum_{w_i \in \mathcal{W}_q} TF_{u,w_i} \times IDF_{w_i}.$$

If *u* is an internal node of the tree, $TF_{u,wi}$ is calculated from index vectors in the child nodes of *u*. If the *u* is a leaf node, $TF_{u,wi}$ is calculated as:

$$TF_{u,w_i} = \frac{TF'_{f,w_i}}{\sqrt{\sum_{w_i \in \mathcal{W}} (TF'_{f,w_i})^2}},$$

Keyword Balanced Binary Tree. The balanced binary tree is widely used to deal with optimization problems [35], [36]. The keyword balanced binary (KBB) tree in our scheme is a dynamic data structure whose node stores a vector *D*. The elements of vector *D* are the normalized TF values. Sometimes, we refer the vector *D* in the node *u* to *Du* for simplicity. Formally, the node *u* in our KBB tree is defined as follows:

$$u = \langle ID, D, P_l, P_r, FID \rangle,$$

where *ID* denotes the identity of node *u*, *Pl* and *Pr* are respectively the pointers to the left and right child of node *u*. If the node *u* is a leaf node of the tree, *FID* stores the identity of a document, and *D* denotes a vector consisting of the normalized TF values of the keywords to the document. If the node *u* is an internal node, *FID* is set to *null*, and *D* denotes a vector consisting of the TF values which is calculated as follows:

$$D[i] = max\{u.P_l \to D[i], u.P_r \to D[i]\}, i = 1, ..., m.$$

The detailed construction process of the tree-based index is illustrated in Section 4, which is denoted as BuildIndexTree(*F*).

3.2 The System and Threat Models

The system model in this paper involves three different entities: data owner, data user and cloud server, as illustrated in Fig. 1.

Data owner has a collection of documents *F* =

*{f*1,*f*2,..,*fn}* that he wants to outsource to the cloud server in encrypted form while still keeping the capability to search on them for effective utilization. In our scheme, the data owner firstly builds a secure searchable tree index *I* from document collection *F*, and then generates an encrypted document collection *C* for *F*. Afterwards, the data owner outsources the encrypted collection *C* and the secure index *I* to the cloud server, and securely distributes the key information of trapdoor generation (including keyword IDF values) and document decryption to the authorized data users.

Besides, the data owner is responsible for the update operation of his documents stored in the cloud server. While updating, the data owner generates the update information locally and sends it to the server.

Data users are authorized ones to access the documents of data owner. With *t* query keywords, the authorized user can generate a trapdoor *TD* according to search control mechanisms to fetch *k* encrypted documents from cloud server. Then, the data user can decrypt the documents with the shared secret key.

Cloud server stores the encrypted document collection *C* and the encrypted searchable tree index *I* for data owner. Upon receiving the trapdoor *TD* from the data user, the cloud server executes search over the index tree *I*, and finally returns the corresponding collection of top- *k* ranked encrypted documents. Besides, upon receiving the update information from the data owner, the server needs to update the index *I* and document collection *C* according to the received information.

The cloud server in the proposed scheme is considered as "honest-but-curious", which is employed by lots of works on secure cloud data search [25], [26], [27]. Specifically, the cloud server honestly and correctly executes instructions in the designated protocol. Meanwhile, it is curious to infer and analyze received data, which helps it acquire additional information. Depending on what information the cloud server knows, we adopt the two threat models proposed by Cao *et al.* [26].

Known Ciphertext Model. In this model, the cloud server only knows the encrypted document collection *C*, the searchable index tree *I*, and the search trapdoor *TD* submitted by the authorized user. That is to say, the cloud server can conduct ciphertext-only attack (COA) [37] in this model.

Known Background Model. Compared with known ciphertext model, the cloud server in this stronger model is equipped with more knowledge, such as the term frequency (TF) statistics of the document collection. This statistical information records how many documents are there for each term frequency of a specific keyword in the whole document collection, as shown in Fig. 2, which could be used as the keyword identity. Equipped with such statistical information, the cloud server can conduct TF statistical attack to deduce or even identify certain keywords through analyzing histogram and value range of the corresponding frequency distributions [24], [25],[27].

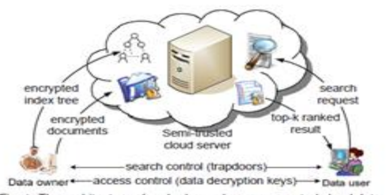

Fig. 1. The architecture of ranked search over encrypted cloud data

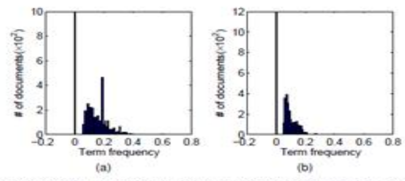

Fig. 2. Distribution of term frequency (TF) for (a) keyword "subnet", and (b) keyword "host".

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multikeyword ranked search over outsourced encrypted cloud data under the above models, our system has the following design goals.

Dynamic: The proposed scheme is designed to provide not only multi-keyword query and accurate result ranking, but also dynamic update on document collections.

Search Efficiency: The scheme aims to achieve sublinear search efficiency by exploring a special tree-based index and an efficient search algorithm.

Privacy-preserving: The scheme is designed to prevent the cloud server from learning additional information about the document collection, the index tree, and the query. The specific privacy requirements are summarized as follows,

1) ***Index Confidentiality and Query Confidentiality:***

The underlying plaintext information, including keywords in the index and query, TF values of keywords stored in the index, and IDF values of query keywords, should be protected from cloud server.

2) *Trapdoor Unlinkability:* The cloud server should not be able to determine whether two encrypted queries (trapdoors) are generated from the same search request.

3) *Keyword Privacy:* The cloud server could not identify the specific keyword in query, index or document collection by analyzing the statistical information like term frequency. Note that our proposed scheme is not designed to protect access pattern, i.e., the sequence of returned documents.

IV. THE PROPOSED SCHEMES

In this section, we firstly describe the unencrypted dynamic multi-keyword ranked search (UDMRS) schemewhich is constructed on the basis of vector space model and KBB tree. Based on the UDMRS scheme, two secure search schemes (BDMRS and EDMRS schemes) are constructed against two threat models, respectively.

4.1 Index Construction of UDMRS Scheme

In Section 3, we have briefly introduced the KBB index tree structure, which assists us in introducing the index construction. In the process of index construction, we first generate a tree node for each document in the collection. These nodes are the leaf nodes of the index tree. Then, the internal tree nodes are generated based

on these leaf nodes. The formal construction process of the index is presented in Algorithm 1. An example of our index tree is shown in Fig. 3. Note that the index tree *T* built here is a plaintext.Following are some notations for Algorithm 1. Besides, the data structure of the tree node is defined as ⟨*ID;D; Pl; Pr; FID*⟩, where the unique identity *ID* for each tree node is generated through the function GenID().

• *CurrentNodeSet* – The set of current processing nodes which have no parents. If the number of nodes is even, the cardinality of the set is denoted as $2h(h \in \mathbf{Z}+)$, else the cardinality is denoted as $(2h + 1)$.

• *TempNodeSet* – The set of the newly generated

nodes.In the index, if $Du[i] \neq 0$ for an internal node *u*, there is at least one path from the node *u* to some leaf, which indicates a document containing the keyword *wi*.

In addition, *Du*[*i*] always stores the biggest normalized TF value of *wi* among its child nodes. Thus, the possible largest relevance score of its children can be easily estimated.

V. CONCLUSION AND FUTURE WORK

In this paper, a secure, efficient and dynamic search scheme is proposed, which supports not only the accurate multi-keyword ranked search but also the dynamic deletion and insertion of documents. We construct a special keyword balanced binary tree as the index, and propose a "Greedy Depth-first Search" algorithm to obtain better efficiency than linear search. In addition, the parallel search process can be carried out to further reduce the time cost. The security of the scheme is protected against two threat models by using the secure kNN algorithm. Experimental results demonstrate the efficiency of our proposed scheme.

There are still many challenge problems in symmetric SE schemes. In the proposed scheme, the data owner is responsible for generating updating information and sending them to the cloud server. Thus, the data owner needs to store the unencrypted index tree and the information that are necessary to recalculate the IDF values. Such an active data owner may not be very suitable for the cloud computing model. It could be a meaningful but difficult future work to design a dynamic searchable encryption scheme whose updating operation can be completed by cloud server only, meanwhile reserving the ability to support multi-keyword ranked search. In addition, as the most of works about searchable encryption, our scheme mainly considers the challenge from the cloud server. Actually, there are many secure challenges in a multi-user scheme. Firstly, all the users usually keep the same secure key for trapdoor generation in a symmetric SE scheme. In this case, the revocation of the user is big challenge. If it is needed to revoke a user in this scheme, we need to rebuild the index and distribute the new secure keys to all the authorized users. Secondly, symmetric SE schemes usually assume that all the data users are trustworthy. It is not practical and a dishonest data user will lead to many secure problems. For example, a dishonest data user may search the documents and distribute the decrypted documents to the unauthorized ones. Even more, a dishonest data user may distribute his/her secure keys to the unauthorized ones. In the future works, we will try to improve the SE scheme to handle these challenge problems.

VI. REFERENCES

[1] K. Ren, C.Wang, Q.Wang *et al.*, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.

[3] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[4] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology- Eurocrypt 2004*. Springer, 2004, pp. 506–522.

[6] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, "Public key encryption that allows pir queries," in *Advances in Cryptology-CRYPTO 2007*. Springer, 2007, pp. 50–67.

[7] 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 on*. IEEE, 2000, pp. 44–55.

[8] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[9] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the Third international conference on Applied Cryptography and Network Security*. Springer-Verlag, 2005, pp. 442–455.

[10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 79–88.

AUTHOR's PROFILE

K.Narendra received B.Tech Degree from St.Johns College of engineering and technology in Kurnnol. He is currently pursuing M.tech Degree in Software Engineering specialization in St.Johns College of engineering and technology, Kurnool, India.

Mr.GKV Narasimha Reddy received Ph.D from Annamalai University and received M.tech degree from Sathyabama University. He is currently working as Associate professor, Department of CSE, in St.Johns College of engineering and technology, Kurnool, Andhra Pradesh, India. His interests include Computer Networks, Operating system, Data Base Management Systems.