A Multi-keywordRanked Search Scheme over Encrypted based on Hierarchical Clustering Index

Indhuja.A
M.E., CSE Scholar,
Annapoorana Engineering College.

T.P.Udhayashankar M.E.,
Associate Professor, Department of CSE,
Annapoorana Engineering College,

A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data 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 project, 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 TFIDF model are combined in the index construction and query generation. The proposed hierarchical approach clusters the documents based on the minimum relevance threshold, and then partitions the resulting clusters into sub-clusters until the constraint on the maximum size of cluster is reached. In the search phase, this approach can reach a linear computational complexity against an exponential size increase of document collection. In order to verify the authenticity of search results, a structure called minimum hash sub-tree is designed in this paper. Due to the use of our special tree-ba
sed 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.

***KeyTerms*—Searchable encryption, multi-keyword ranked search, dynamic update, cloud computing.**

1.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 datathemselves.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, whichare 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.

In order to address the above problem, researcher-s have designed some general-purpose solutions with fully-homomorphic encryption [3] or oblivious RAMs [4]. However, these methods are not practical due to their high computational overhead for both the cloud sever and user.

On the contrary, more practical special-purpose solutions, such as searchable encryption (SE) schemes have made specific contributions in terms of efficiency, functionality and security. Searchable encryption schemes enable the client to store the encrypted data to the cloud and execute keyword search over cipher text domain. So far, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search, multi-keyword Boolean search, ranked search, multi-keyword ranked search, etc. Among them, multi-keyword ranked search achieves more and more attention for its practical applicability.

Recently, some dynamic schemes have been proposed to support inserting and deleting operations on document collection. These are significant works as it is highly possible that the data owners need to update their data on the cloud server. But few of the dynamic schemes support efficient multi-keyword ranked search. 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 in-dex 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 vector. 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 "Hierarchical clustering" algorithm. Moreover, parallel search can be flexibly performed to further reduce the time cost of search process.

2 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 or symmetric key based cryptography .

The first symetric searchable encryption (SSE) scheme, and the search time of their scheme is linear to the size of the data collection. 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. 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. Afterward, abundant works have been proposed under different threat models to achieve various search functionality, such as single keyword search, similarity search, multi-keyword Boolean search, ranked search and multi-keyword ranked search etc.

Multi-keyword Boolean search allows the users to input multiple query keywords to request suitable documents. Among these works, conjunctive keyword search schemes only return the documents that contain all of the query keywords. Disjunctive keyword search schemes return all of the documents that contain a subset of the query keywords. Predicate search schemes 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 have realized the ranked search using order-preserving techniques, but they are designed only for single keyword search. 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, 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. 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 search algorithm achieves better-than-linear search efficiency but results in precision loss. 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 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 relevant search results. However, these works don't support dynamic operations.

Practically, the data owner may need to update the document collection after he upload the collection to the cloud server. Thus, the SE schemes are expected to sup-port the insertion and deletion of the documents. There are also several dynamic searchable encryption schemes. In the work of the each document is considered as a sequence of fixed length words, and is individually indexed. This scheme supports straight-forward update operations but with low efficiency. A scheme to generate a sub-index (Bloom filter) for every document based on keywords. Then the dynamic operations can be easily realized through updating of a Bloom filter along with the corresponding document. However, Goh's scheme has linear search time and suffers from false positives. In 2012, constructed an encrypted inverted index thatcan handle dynamic data efficiently. But, this scheme is very complex to implement. Subsequently, as an improvement, A new search scheme based on tree-based index, which can handle dynamic update on document data stored in leaf nodes.

However, their scheme is designed only for single-keyword Boolean search. In A data structure for keyword/identity tuples named "T-Set". Then, a document can be represented by a series of independent T-Sets. Based on this structure, proposed a dynamic searchable encryption scheme. In their construction, newly added tuples are stored in another database in the cloud, and deleted tuples are recorded in a revocation list.The final search result is achieved through excluding tuples in the revocation list from the ones retrieved from original and newly added tuples. Yet, Cash *et al.*'s dynamic search scheme doesn't realize the multi-keyword ranked search functionality.

3 PROBLEM FORMULATION

Notations and Preliminaries

- W – The dictionary, namely, the set of keywords,denoted as $W = \{w_1; w_2; :::; w_m\}$.
- m – The total number of keywords in W.

- W_q– The subset of W, representing the keywordsin the query.
- F – The plaintext document collection, denoted as acollection of n documents $F = \{f_1; f_2; :::; f_n\}$. 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 thecloud server, denoted as $C = \{c_1; c_2; :::; c_n\}$.
- T – The unencrypted form of index tree for thewhole document collection F.
- I – The searchable encrypted tree index generatedfrom T .

- Q – The query vector for keyword set W_q.
- TD – The encrypted form of Q, which is named astrapdoor for the search request.
- D_u– The index vector stored in tree node u whosedimension equals to the cardinality of the dictionary W. Note that the node u can be either a leaf nodeor an internal node of the tree.

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 .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 ofquery 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 D_u and the IDF vector Q can be calculated to quantify the relevance between the query and corresponding document.

3.2 The System Models

The system model in this paper involves three different entities: data owner, data user and cloud server,

Data owner has a collection of documents$F=\{f_1; f_2; :::; f_n\}$ that he wants to outsource to the cloudserver 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 datausers.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 dataowner. 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 receivingthe 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 Specif-ically, 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.

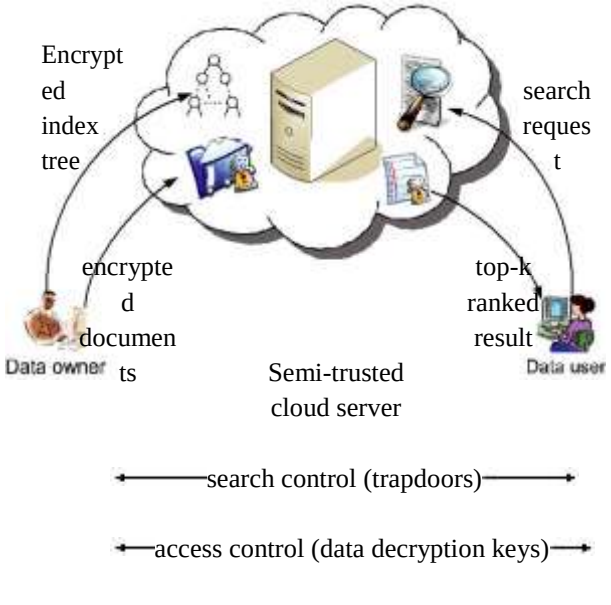
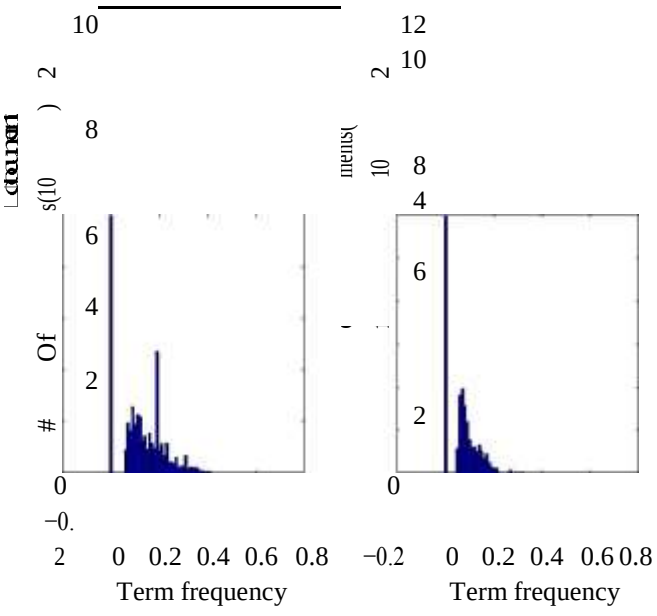


Fig. 1. The architecture of ranked search over encrypted cloud data



(a) (b)

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

(b) keyword "host".

Known Cipher text Model. In this model, the cloudserver 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, thecloud server can conduct cipher text-only attack (COA) in this model.

Known Background Model. Compared with knowncipher text 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

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multi-keyword 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 pro-vide not only multi-keyword query and accurate result ranking, but also dynamic update on document collections.

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

Privacy-preserving: The scheme is designed to pre-vent 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:* Theunderlying plaintext information, including key-words in the index and query, TF values of key-words stored in the index, and IDF values of query keywords, should be protected from cloud server;

2) *Trapdoor Unlink ability:* The cloud server shouldnot 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 document.

4 THE PROPOSED SCHEMES

A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data We explore the problem of maintaining the semantic relationship between different plain documents over the related encrypted documents and give the design method to enhance the performance of the scemantic search. We also propose the Hierarchical Clustering Algorithm to adapt to the requirements of data explosion, online information retrieval and semantic search. At the same time, a Verifiable mechanism is also proposed to guarantee the correctness and

completeness of search results. Built to evaluate the search efficiency, accuracy, and rank security.

The experiment result proves that the proposed architecture not only properly solves the multi-keyword ranked search problem, but also brings an improvement in search efficiency, rank security, and the relevance between retrieved documents.

4.1 PROPOSED SYSTEM ALGORITHM

4.1.1 Hierarchical Clustering Algorithm

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. The quality of a pure hierarchical clustering method suffers from its inability to perform adjustment, once a merge or split decision has been executed. Then it will neither undo what was done previously, nor perform object swapping between clusters. Thus merge or split decision, if not well chosen at some step, may lead to some-what low-quality clusters.

One promising direction for improving the clustering quality of hierarchical methods is to integrate hierarchical clustering with other techniques for multiple phase clustering.Describe a few improved hierarchical clustering algorithms that overcome the limitations that exist in pure hierarchical clustering algorithms.

Algorithm *Quality Hierarchical Clustering (QH*

1, input documents and set the size threshold *TH*
2, build cluster set C_0 in first level by *dynamic k-mean*
3, **while** there are new cluster set C_i
4, **for** every cluster $C_{i,j}$
5, **if** the size of $C_{i,j}$ is bigger than *TH*
6, split this cluster into sub-clusters C_{i+1}
7, **Until** all clusters match the size constraint

Cluster Analysis (data segmentation) has a variety of goals that relate to grouping or segmenting a collection of objects (i.e., observations, individuals, cases, or data rows) into subsets or clusters, such that those within each cluster are more closely related to one another than objects assigned to different clusters. Central to all of the goals of cluster analysis is the notion of degree of similarity (or dissimilarity) between the individual objects being clustered.

There are two major methods of clustering: hierarchical clustering and k-means clustering. For information on k-means clustering, refer to the k-Means Clustering section. In hierarchical clustering, the data is not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to n clusters that each contains a single object.

Hierarchical Clustering is subdivided into agglomerative methods, which proceed by a series of fusions of the n objects into groups, and divisive methods, which separate n objects successively into finer groupings. Agglomerative techniques are more commonly used, and this is the method implemented in XLMiner. Hierarchical clustering may be represented by a two-dimensional diagram known as a dendrogram, which illustrates the fusions or divisions made at each successive stage of analysis. Following is an example of a dendrogram.

An agglomerative hierarchical clustering procedure produces a series of partitions of the data, P_n, P_{n-1}, P_1. The first P_n consists of n single object clusters, the last P_1, consists of single group containing all n cases. At each particular stage, the method joins together the two clusters that are closest together (most similar). (At the first stage, this amounts to joining together the two objects that are closest together, since at the initial stage each cluster has only one object.)

Differences between methods arise due to different ways of defining distance (or similarity) between clusters. The following sections describe several agglomerative techniques in detail. In the complete linkage, also called farthest neighbor, the clustering method is the opposite of single linkage. Distance between groups is now defined as the distance between the most distant pair of objects, one from each group. One of the simplest agglomerative hierarchical clustering methods is single linkage, also known as the nearest neighbor technique. The defining feature of the method is that distance between groups is defined as the distance between the closest pair of objects, where only pairs consisting of one object from each group are considered

4.2 ENCRYPTED CLOUD DATA ARCHITECTURE

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

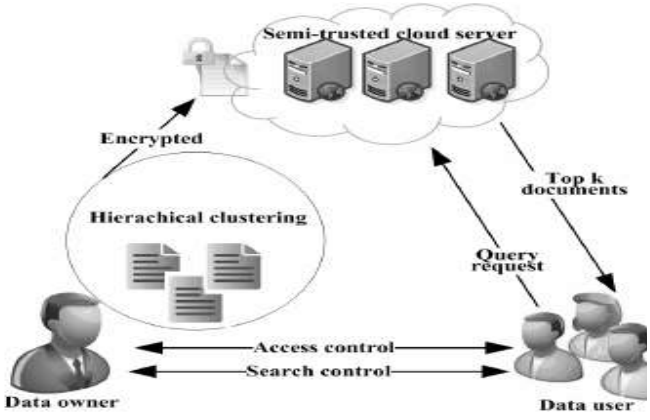


Fig. 4.2: Architecture of Multi keyword Ranked Search over Encrypted Cloud Data

In this project 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. 1; f , 2; f n 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 serverstores 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 topk 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. We analyze the BDMRS scheme according to the three predefined privacy requirements in the design goals,

Index Confidentiality and Query Confidentiality: In the proposed BDMRS scheme, Iand TD are obfuscated vectors, which mean the cloud server cannot infer the original vectors Duand Q without the secret key set SK. The secret keys Mu1and Mare Gaussian random matrices. According to the attacker (cloud server) of COA cannot calculate the matrices merely with cipher text. Thus, the BDMRS scheme is resilient against cipher text-only attack (COA) and the index confidentiality and the query confidentiality are well protected.

Query Unlink ability: The trapdoor of query vector is generated from a random splitting operation, which means that the same search requests will be transformed into different query trapdoors, and thus the query unlink ability is protected. However, the cloud server is able to link the same search requests according to the same visited path and the same relevance scores.

Keyword Privacy: In this scheme, the confidentiality of the index and query are well protected that the original vectors are kept from the cloud server. And the search process merely introduces inner product computing of encrypted vectors, which leaks no information about any specific keyword. Thus, the keyword privacy is protected in the known cipher text model. But in the known background model, the cloud server is supposed to have more knowledge, such as the term frequency statistics of keywords.

This statistic information can be visualized as a TF distribution histogram which reveals how many documents are there for every TF value of a specific keyword in the document collection. Then, due to the specificity of the TF distribution histogram, like the graph slope and value range, the cloud server could conduct TF statistical attack to deduce/identify keywords.

In the worst case, when there is only one keyword in the query vector, i.e. the normalized IDF value for the keyword is 1, the final relevance score distribution is exactly the normalized TF distribution of this keyword, which is directly exposed to cloud server. Therefore, the BDMRS scheme cannot resist TF statistical attack in the known background model.

5 RESULT AND ANALYSIS

The retrieved data have high possibility to be wrong since the network is unstable and the data may be damaged due to the hardware/software failure or malicious administrator or intruder. Verifying the authenticity of search results is emerging as a critical issue in the cloud environment. We, therefore, designed a signed hash tree to verify the correctness and freshness of the search results. The data owner builds the hash tree basedon the hierarchical index structure. The hash value of the leaf node of the tree is hðid k version k FðidÞÞ where id means document id, version means document version and FðidÞ means the document contents. The value of non-leaf node is a pair of values ðid; hðid k hÞÞ where id denotes the value of the cluster center or document vector in the encrypted index and hchildis the hash value of its child node.

The hash value of tree root node is based on the hash values of all clusters in the first level. It is worth noting that the root node denotes the data set which contains all clusters. Then the data owner generates the signature of the hash values of the root node and outsources the hash tree including the root signature to the cloud server. Cryptographic signature s (e.g., RSA signature, DSA signature) can be used here to authenticate the hash value of root node of ChildProcessing. By the algorithm shown in the Fig. 9, the cloud server returns the root signature and the minimum hash sub-tree (MHST) to client. The minimum hash sub-tree includes the hash values of leaf nodesin the matched cluster and non-leaf node corresponding to all cluster centers used to find the matched cluster in the searching phase. The search result is document D, E and F. Then the leaf nodes are D, E, F and G, and non leaf nodes includes C1, C2, C3, C4, dD, dE, d. In addition, the root is included in the non-leaf node Verifying.

The data owner uses the minimum hash sub-tree to re-compute the hash values of nodes, in particular the root node which can be further verified by the root signature. If all nodes are matched, then the correctness and freshness is guaranteed. Then the data owner re-searches the index constructed by retrieved values in MHST.

If the search result is same as the retrieved result, the completeness, correctness and freshness all are guaranteed. As the documents stored at server may be deleted or modified and new documents may be added to the original data collection, a mechanism which supports dynamic data collection is necessary. A naive way to address these problems is downloading all documents and index locally and updating the data collection and index. However, this method needshuge cost in bandwidth and local storage space. To avoid updating index frequently, we provide a practical strategy to deal with insertion, deletion and modification operations. Without loss of generality, we use following examples to illustrate the workings of the strategy.

The data owner preserves many empty entries in the dictionary for new documents. If a new document contains new keywords, the data owner first adds these new keywords to the dictionary and then constructs a document vector based on the new dictionary. The data owner sends the trapdoor generated by the document vector; encrypted document and encrypted document vector to the cloud sever. The cloud severs find the closest cluster, and puts the encrypted document and encrypted document vector into it.As every cluster has a constraint on the maximum size, it is possible that the number of documents in a cluster exceeds the limitation due to the insertion operation. In this case, all the encrypted document vectors belonging to the broken cluster are returned to the data owner.

After decryption of the retrieved document vectors, the data owner re-builds the sub-index based on the deciphered document vectors.

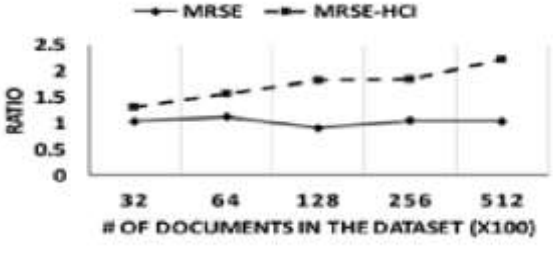
The sub-index is re-encrypted and re-outsourced to the cloud server.Upon receiving a deletion order, the cloud server searches the target document. Then the cloud server deletes the document and the corresponding document vector.Modifying a document can be described as deleting the old version of the document and inserting the new version. The operation of modifying documents, therefore, can be realized by combining insertion operation and deletion operation.

To deal with this impact on the hash tree, a lazy update strategy is designed. For the insertion operation, the corresponding hash value will be calculated and marked as a raw node, while the original nodes in the hash tree will be kept unchanged because the original hash tree still supports document verification except the new document.

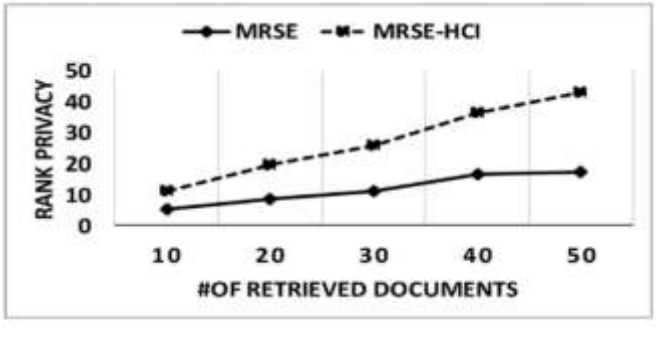
Only when the new added document is accessed, the hash tree will be updated. Similar concept is used in the deletion operation. The only difference is that the deletion operation will not bring the hash tree update.

In order to test the performance of MRSE-HCI on real dataset, we built an experimental platform to test the search efficiency, accuracy and rank privacy. We implemented the target experiment based on a distributed platform which includes three Think Server RD830 and a Think Center M8400t.The data set is built from IEEE Xplore, including about 51,000 documents and 22,000 keywords. According to the notations n denotes the dictionary size, k denotes the number of top-k documents, m denotes the number of documents in the data set and w denotes the number of keywords in the users query.

Search efficiency using the different size of document set with unchanged dictionary size number of retrieved documents and number of query keywords, n ¼ 22;157;k ¼ 20;w¼ 5. In Fig. 11b, we adjust the value of k with unchanged dictionary size, document set size and number of query keywords, n ¼ 22; 157; m ¼ 51; 312; w¼ 5. The different number of query keywords with unchanged dictionary size, document set size and number of retrieved documents, n ¼ 22; 157;m ¼ 51; 312; k ¼ 20. Observe that with the exponen-trial growth of document set size, the search time of MRSE increases exponentially, while the search time of MRSE _ HCI increases linearly.



Graph No. 5. 1: Relavance of Document



Graph No. 5.2: Ranked Privacy

The relevance between query and retrieved documents in MRSE-HCI is slightly lower than that in MRSE. Especially, this gap narrows when the data size increases since a big document data set has a clear category distribution which improves the relevance between query and documents. The rank accuracy according to the tradeoff parameter a is set to 1, which means there is no bias towards relevance of documents or relevance between documents and query. From the result, we can conclude that MRSE-HCI is better than MRSE inrank accuracy. The rank privacy according tothis test, no matter the number of retrieved documents, MRSE _ HCI has better rank privacy than MRSE.

This mainly caused by the relevance of documents introduced into search strategy. In this project, we investigated cipher text search in the scenario of cloud storage. Explore the problem of maintaining the semantic relationship between different plain documents over the related encrypted documents and give the design method to enhance the performance of the semantic search. Further propose the MRSE-HCI architecture to adapt to the requirements of data explosion, onlineinformation retrieval and semantic search.

At the sametime, a variable mechanism is also proposed to guaranteethe correctness and completeness of search results. In addition, we analyze the search efficiency and security under two popular threat models. An experimental platform is built to evaluate the search efficiency, accuracy, and rank security.

The experiment result proves that the proposed architecture not only properly solves the multi-keyword ranked search problem, but also brings an improvement in search efficiency, rank security, and the relevance between retrieved documents.

6 CONCLUSION AND FUTURE WORK

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 “Hierarchical Clustering” 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.

REFERENCES

[1]. Zhihua Xia, Xinhui Wang, Xingming sun, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme Over Encrypted Cloud Data" vol. 16, no. 1, pp. 69–73, 2015.

[2]. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in INFOCOM, 2010 Proceedings IEEE. IEEE, 2010.

[3]. Y. H. Hwang and P. J. Lee, "Ranking Based Personalized Search Engine using Android," in Proceedings of the First international conference on Pairing-Based Cryptography. Springer-Verlag, 2007.

[4]. P. Golle, J. Staddon, and B. Waters, "Storing Shared Data on the Cloud via Security- Ediator," in Applied Cryptography and Network Security. Springer, 2004.

[5]. 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.

[6]. 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.

[7]. B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi keyword fuzzy search over encrypted data in the cloud," in IEEE INFOCOM, 2014.

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