IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

The Encrypted Data Over the Cloud In A Secure And Dynamic Multi-Keyword Search Ranking Scheme

MS.C.ANUSHA MR.N.MANEIAH

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 provideefficient 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 .

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 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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

• Zhihua Xia, Xinhui Wang, and Xingming Sun are with the Jiangsu Engineering Center of Network Monitoring, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, and School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, China. E-mail: xia zhihua, wxh nuist, sunnudt@163.com. • Qian Wang is with the the School of Computer, Wuhan University, Wuhan, China. E-mail: qianwang@whu.edu.cn.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. In order to address the above problem, researchers 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 specialpurpose 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 ciphertext 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, multikeyword 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

cloud server. But few of the dynamic schemes support efficient multikeyword ranked search. This paper proposes a secure tree-based search scheme over the encrypted cloud data, which supports multikeyword ranked search and dynamic operation on the document collection. Specifically, the vector space model andthewidely-used"termfrequency(TF)×inversedocument frequency (IDF)" model are combined in the index construction and query generation to provide multikey word 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 ciphertext 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 kepttologarithmic .Andinpractice, 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. The reminder of this paper is organized as follows. Related work is discussed in Section 2, and Section 3 gives a brief introduction to the system model, threat model, the design goals, and the preliminaries. Section 4 describes the schemes in detail. Section 5 presents the

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

experiments and performance analysis. And Section 6 covers the conclusion.

2 RELATED WORK

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keyword search over ciphertext domain. Due to different cryptography primitives, searchable encryption schemes can be constructed using public key based cryptography or symmetric key based cryptography Song et al. 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 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. 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 multikeyword 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 documentscaneffectivelydecreasenetworktra

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

ffic.Some early works have realized the ranked search using order-preserving techniques, but they are designed only for single keyword search. Cao et al 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. 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. ¨Orencik et al] 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 Zhang et al. proposed a scheme to deal with secure multi-keywordrankedsearchinamulti-ownermodel.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 support the insertion and deletion of the documents. There are also several dynamic searchable encryption schemes. In the work of Song et al. [7], the each document is considered as a sequence of fixed length words, and is individually indexed.

3 PROBLEM FORMULATION

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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: RScore(Du,Q) = Du ·Q = ∑ wi∈Wq TFu,wi ×IDFwi. (1) If u is an internal node of the tree, TFu,wi is calculated from index vectors in the child nodes of u. If the u is a leaf node, TFu,wi is calculated as:

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 = {f1,f2,...,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. Cloudserver 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 topk ranked encrypted documents. Besides, upon receiving the update information from the data owner, the

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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 .

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multikeyword ranked searchoverout sourced 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.

4 THE PROPOSED SCHEMES

In this section, we firstly describe the unencrypted dynamic multi-keyword ranked search (UDMRS) scheme which 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.

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

4.1 Index Construction of UDMRS Scheme

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⟩,wheretheuniqueidentityIDf oreach 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 ∈ Z+), else the cardinality is denoted as (2h + 1). • TempNodeSet – The set of the newly generated nodes. In the index, if Du[i]≠ 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.

4.2 Search Process of UDMRS Scheme The search process of the UDMRS scheme is a recursive procedure upon the tree, named as "Greedy Depthfirst Search (GDFS)" algorithm. We construct a result list denoted as RList, whose element is defined as ⟨RScore,FID⟩. Here, the RScore is the relevance score of the document fFID to the query, which is calculated accordingtoFormula(1).The RList storesthe k accessed documents with the largest relevance scores to the query. The elements of the list are ranked in descending order according to the RScore, and will be updated timely during the search process. Following are some other notations, and the GDFS algorithm is described in Algorithm 2. • RScore(Du,Q) – The function to calculate the relevance score for query vector Q and index vector Du stored in node u, which is

defined in Formula (1). • kthscore – The smallest relevance score in current RList, which is initialized as 0. • hchild – The child node of a tree node with higher relevance score.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2401003, IEEE Transactions on Parallel and Distributed Systems

Algorithm 1 BuildIndexTree(F) Input: the document collection F = {f1,f2,...,fn} with the identifiers FID = {FID|FID = 1,2,...,n}. Output: the index tree T 1: for each document fFID in F do 2: Construct a leaf node u for fFID, with u.ID = GenID(), u.Pl = u.Pr = null, u.FID = FID, and D[i] = TFfFID,wi for i = 1,...,m;— 3: Insert u to CurrentNodeSet; 4: end for 5: while the number of nodes in CurrentNodeSet is larger than 1 do 6: if the number of nodes in CurrentNodeSet is even, i.e. 2h then 7: for each pair of nodes u′ and u″ in CurrentNodeSet do 8: Generate a parent node u for u′ and u″, with u.ID = GenID(), u.Pl = u′, u.Pr = u″, u.FID = 0 and D[i] = max{u′.D[i],u″.D[i]} for each i = 1,...,m; 9: Insert u to TempNodeSet; 10: end for 11: else 12: for each pair of nodes u′ and u″ of the former (2h−2) nodes in CurrentNodeSet do13: Generate a parent node u for u′ and u″; 14: Insert u to TempNodeSet; 15: end for 16: Create a parent node u1 for the (2h−1)-th and 2h-th node, and then create a parent node u for u1 and the (2h + 1)-th node; 17: Insert u to TempNodeSet; 18: end if 19: Replace CurrentNodeSet with TempNodeSet and then clear TempNodeSet; 20: end while 21: return the only node left in CurrentNodeSet, namely, the root of index tree T;

Algorithm 2 GDFS(IndexTreeNode u) 1: if the node u is not a leaf node then 2: if RScore(Du,Q) > kthscore then 3:

GDFS(u.hchild); 4: GDFS(u.lchild); 5: else 6: return 7: end if 8: else 9: if RScore(Du,Q) > kthscore then 10: Delete the element with the smallest relevance score from RList; 11: Insertanewelement⟨RScore(Du,Q),u.FID⟩an d sort all the elements of RList; 12: end if 13: return 14: end if

• lchild – The child node of a tree node with lower relevance score. Sincethepossiblelargestrelevancescoreofdoc uments rooted by the node u can be predicted, only a part of the nodes in the tree are accessed during the search process. Fig. 3 shows an example of search process with the document collection F = {fi|i = 1,...,6}, cardinality of the dictionary m = 4, and query vector Q = (0,0.92,0,0.38).

4.3 BDMRS Scheme Based on the UDMRS scheme

we construct the basic dynamic multi-keyword ranked search (BDMRS) scheme by using the secure kNN algorithm [38]. The BDMRS scheme is designed to achieve the goal of privacypreserving in the known ciphertext model, and the four algorithms included are described as follows: • SK←Setup()

Initially,thedataownergeneratesthe secret key set SK, including 1) a randomly generated m-bit vector S where m is equal to the cardinality of dictionary, and 2) two (m×m) invertible matrices M1 and M2. Namely, SK = {S,M1,M2}. • I ← GenIndex(F,SK) First, the unencrypted index tree T is built on F by using T ← BuildIndexTree(F). Secondly, the data ownergeneratestworandomvectors{Du′,Du″ }forindexvector Du in each node u, according to the secret vector S. Specifically, if S[i] = 0, Du′[i] and Du″[i] will be set equal to Du[i]; if S[i] = 1, Du′[i] and Du″[i] will be set as two random values whose sum equals to Du[i]. Finally, the encrypted index tree I is built wherethenode u storestwoencryptedindexvectors Iu = {MT 1 Du′,MT 2 Du″}. • TD←GenTrapdoor(Wq,SK) With keyword set Wq, the unencrypted query vector Q with length of m is generated. If wi ∈Wq, Q[i] stores the normalized IDF value of wi; else Q[i] is set to 0. Similarly, the query vector Q is split into two random vectors Q′ and Q″. The difference is that if S[i] = 0, Q′[i] and

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

Q″[i] are set to two random values whose sum equals to Q[i]; else Q′[i] and Q″[i] are set as the same as Q[i]. Finally, the algorithm returns the trapdoor TD = {M−1 1 Q′,M−1 2 Q″}. • RelevanceScore ← SRScore(Iu,TD) With the trapdoor TD, the cloud server computes the relevance score of node u in the index treeI to the query. Note that the relevance score calculated from encrypted vectors is equal to that from unencrypted vectors as follows: Iu ·TD = (MT 1 Du′)·(M−1 1 Q′) + (MT 2 Du″)·(M−1 2 Q″) = (MT 1 Du′)T(M−1 1 Q′) + (MT 2 Du″)T(M−1 2 Q″) = Du′TM1M−1 1 Q′ + Du″TM2M−1 2 Q″ = Du′·Q′ + Du″·Q″ = Du ·Q = RScore(Du,Q) (6)

Security analysis. We analyze the BDMRS scheme according to the three predefined privacy requirements in the design goals: 1) Index Confidentiality and Query Confidentiality: In the proposed BDMRS scheme, Iu and TD are obfuscated vectors, which means the cloud server cannot infer the original vectors Du and Q without the secret key set SK. The secret keys M1 and M2 are Gaussian random matrices. According to [38], the attacker (cloud server) of COA cannot calculate the matrices merely with ciphertext. Thus, the BDMRS scheme is resilient against ciphertext-only attack (COA) and the index confidentiality and the query confidentiality are well protected. 2) Query Unlinkability: 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 unlinkability 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. 3) Keyword Privacy: In this scheme, the confidentiality of the index and query are well protected that the originalvectorsarekeptfromthecloudserver.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 ciphertext 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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 .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.

4.4 EDMRS Scheme

The security analysis above shows that the BDMRS scheme can protect the Index Confidentiality and Query Confidentiality in the known ciphertext model. However, the cloud server is able to link the same search requests by tracking path of visited nodes. In addition, in the known background model, it is possible for the cloud server to identify a keyword as the normalized TF distribution of the keyword can be exactly obtained from the final calculated relevance scores. The primary cause is that the relevance score calculated from Iu and TD is exactly equal to that from Du and Q. A heuristic method to further improve the security is to break such exact equality. Thus, we can introduce some tunable randomness to disturb the relevance score calculation. In addition, to suit different users' preferences for higher accurate ranked results or better protected keyword privacy, the randomness are set adjustable.

4.5 Dynamic Update Operation of DMRS

After insertion or deletion of a document, we need to update synchronously the index. Since the index of DMRS scheme is designed as a balanced binary tree, the dynamic operation is carried out by updating nodes in the index tree. Note that the update on index is merely based on document identifies, and no access to the content of documents is required. The specific process is presented as follows: • {I′ s,ci} ← GenUpdateInfo(SK,Ts,i,updtype)) This

algorithm generates the update information {I′ s,ci} which will be sent to the cloud server. In order to reduce the communication overhead, the data owner stores a copy of unencrypted index tree. Here,thenotion updtype ∈{Ins,Del}denoteseither an insertion or a deletion for the document fi. The notionTs denotes the set consisting of the tree nodes that need to be changed during the update. For example, if we want to delete the document f4 in Fig. 3, the subtree Ts includes a set of nodes {r22,r11,r}. – If updtype is equal to Del, the data owner deletesfromthe subtreethe leaf node that stores the document identity i and updates the vector D of other nodes in subtreeTs, so as to generate the updated subtree T′ s. In particular, if the deletion of the leaf node breaks the balance of the binary index tree, we replace the deleted node with a fake node whose vector is padded with 0 and file identity is null. Then, the data owner encrypts the vectors stored in the subtree T′ s with the key set SK to generate encrypted subtree I′ s, and set the output ci as null. – If updtype is equal to Ins, the data owner generates a tree node u = ⟨GenID(),D,null,null,i⟩ for the document fi, where D[j] = TFfi,wj for j = 1,...,m. Then, the data owner inserts this new node into the subtreeTs as a leaf node and updates the vector D of other nodes in subtree Ts according to the Formula (5), so as to generate the new subtree T′ s. Here, the data owner is always preferable to replace the fake leaf nodes generated by Del operation with newly inserted nodes, instead of directly inserting new nodes. Next,thedataownerencryptsthevectorsstored in subtree T′ s with the key set SK as described in Section 4.4, to generate encrypted subtree I′ s. Finally, the document fi is encrypted to ci. • {I′,C′} ← Update(I,C,updtype,I′ s,ci) In this algorithm, cloud server replaces the corresponding subtree Is(the encrypted form of Ts) with I′ s, so as to generate a new index tree I′. If updtype is equal to Ins, cloud server inserts the encrypted document ci into C, obtaining a new collection C′. If updtype is equal to Del, cloud server deletes the encrypted document ci from C to obtain the new collection C′. Similar to the scheme in [31],

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

our scheme can also carry out the update operation without storing the index

4.6 Parallel Execution of Search

Owing to the tree-based index structure, the proposed search scheme can be executed in parallel, which further improves the search efficiency. For example, we assume there are a set of processors P = {p1,...,pl} available. Given a search request, an idle processor pi is used to query the root r. If the search could be continued on both the children, and there is an idle processor pj, the processor pi continues to deal with one of the children while processor pj deals with the other one. If there is no idle processor, the current processor is used to deal with the child with larger relevance score, and the other child is put into a waiting queue. Once there is an idle processor, it takes the oldest node in the queue to continue the search. Note that all the processors share the same result list RList.

5 PERFORMANCEANALYSIS

We implement the proposed scheme using C++language in Windows 7 operation system and test its efficiency on a real-world document collection: the Request for Comments (RFC) [39]. The test includes 1) the search precision on different privacy level, and 2) the efficiency of index construction, trapdoor generation, search, and update. Most of the experimental results are obtained with anIntel Core(TM)Duo Processor(2.93 GHz), except that the efficiency of search is tested on a server with two Intel(R) Xeon(R) CPU E5-2620 Processors (2.0 GHz), which has 12 processor cores and supports 24 parallel threads.

5.1 Precision and Privacy

The search precision of scheme is affected by the dummy keywords in EDMRS scheme. Here, the 'precision' is defined as that in [26]: Pk = k′/k, where k′ is the number of real top-k documents in the retrieved k documents. If a smaller standard deviation σ is set for the random document collection with the fixed dictionary, m = 4000, and (b) for the different sizes of dictionary with the fixed document collection ,n = 1000.

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

5.2 Efficiency

5.2.1 Index Tree Construction

The process of index tree construction for document collection F includes two main steps: 1) building an unencrypted KBB tree based on the document collection F, and 2) encrypting the index tree with splitting operation and two multiplications of a (m×m) matrix. The index structure is constructed following a post order traversal of the tree based on the document collection F, and O(n) nodes are generated during the traversal. For each node, generation of an index vector takes O(m) time, vector splitting process takes O(m) time, and two multiplications of a (m×m) matrix takes O(m2) time. As a whole, the time complexity for index tree construction is O(nm2). Apparently, the time cost for building index tree mainly depends on the cardinality of document collection F and the number of keywords in dictionary W. Fig. 5 shows that the time cost of index tree construction is almost linear with the size of document collection, and is proportional to the number of keywords in the dictionary. Due to the dimension extension, the index tree construction of EDMRS scheme is slightly more time-consuming than that of BDMRS scheme. Although the index tree construction consumes relatively much time at the data owner side, it is noteworthy that this is a one-time operation. On the other hand, since the underlying balanced binary tree has space complexity O(n) and every node stores two m-dimensional vectors, the space complexity of the index tree is O(nm). As listed in Table 3, when the document collection is fixed (n = 1000), the storage consumption of the index tree is determined by the size of the dictionary.

5.2.2 Trapdoor Generation

The generation of a trapdoor incurs a vector splitting operation and two multiplications of a (m×m) matrix, thus the time complexity is O(m2), as shown in Fig. 6(a). Typical search requests usually consist of just a few keywords. Fig. 6(b) shows that the number of query keywords has little influence on the overhead of trapdoor generation when the dictionary size is fixed. Due to the dimension extension.

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

5.2.3 Search Efficiency

During the search process, if the relevance score at node u is larger than the minimum relevance score in result list RList, the cloud server examines the children of the node; else it returns. Thus, lots of nodes are not accessed during a real search. We denote the number of leaf nodes that contain one or more keywords in the query as θ. Generally, θ is larger than the number of required documents k, but far less than the cardinality of the document collection n. As a balanced binary tree, the height of the index is maintained to be logn, and the complexity of relevance score calculation is O(m). Thus,

the time complexity of search is O(θmlogn). Note that the real search time is less than θmlogn. It is because 1) many leaf nodes that contain the queried keywords are not visited according to our search algorithm, and 2) the accessing paths of some different leaf nodes share the mutual traversed parts. In addition, the parallel execution of search process can increase the efficiency a lot.

5.2.4 Update Efficiency

In order to update a leaf node, the data owner needs to update logn nodes. Since it involves an encryption operation for index vector at each node, which takes O(m2) time, the time complexity of update operation is thus O(m2 logn). We illustrate the time cost for the

6 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

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

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 securekeys toall the authorizedusers. 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.

REFERENCES

[1] K.Ren,C.Wang,Q.Wangetal.,“Securitychallenges forthepublic 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.

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

[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 CryptologyEurocrypt 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.

[11] 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, pp. 1–5.

[12] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in Data Engineering (ICDE), 2012 IEEE 28th International Conference on. IEEE, 2012, pp. 1156–1167.

[13] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 451–459.

IJMTARC – VOLUME – IV – ISSUE - 16 - DEC 2016 ISSN: 2320-1363

Author's Details

Mr.N.Maneiah received M.Tech(CSE) Degree from School of Information Technology, Autonomous, and Affiliated to JNTUA, Anthapur. He is currently working as Assistant Professor in the Department of Computer Science and Engineering in Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP India. His interests include Object Oriented Programming, Operating System, Database Management System, Computer Networking, Cloud Computing and Software Quality Assurance.

Ms.C.ANUSHA She is currently pursuing M.tech Degree in Computer Science and Engineering specialization in Modugula Kalavathamma Institute of Technology for Women, Rajampet, Kadapa,AP