ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

A Dynamic and Secure Multi-Keyword Ranked Search Scheme over Encrypted Cloud

T.Selvi[1], Santosh[2], Mohammed Arshad [3], S. Rajesh[4]
[1]Assistant professor, [2,3,4]UG Scholars, Computer science, Kings Engineering College, Chennai, India

Abstract:
Due to the increasing popularity of cloudcomputing, more and more data owners are motivated to outsource their datato cloud servers for greatconvenience 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 presenta secure multi-keyword ranked search scheme over encrypted cloud data, which simultaneously supports dynamic update operationslike 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-firstSearch" algorithm to provide efficient multi-keyword ranked search. The secure kNN algorithm is utilized to encrypt the index and queryvectors, and meanwhile ensure accurate relevance score calculation between encrypted index and query vectors. In order to resiststatistical attacks, phantom terms are added to the index vector for blinding search results. Due to the use of our special tree-basedindex 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.

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 . 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.

However, this will cause a huge cost in terms of datausability. For example, the existing techniqueson keyword-based information retrieval, which are widely used on theplaintext 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 or oblivious RAMs. However, these methods are not practical due to their high computational overhead for both the cloud sever and user.

2. LITERATURE SURVEY

2.1 RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems.

Description: MapReduce-based data warehouse systems are playing important roles of supporting big data analytics to understand quickly the dynamics of user behavior trends and their needs in typical Web service providers and social network sites (e.g., Facebook).

Advantage:A fast and space-efficient data placement structure is very important to big data analytics in large-scale distributed systems. The structure are 1) fast data loading, 2) fast query processing, 3) highly efficient storage space utilization, and 4) strong adaptively to highly dynamic workload patterns. Our solution RCFile is designed to meet all the four goals.

Disadvantage: The major weaknesses of row-store for read-only data warehouse systems have been intensively discussed. First, rowstore cannot provide fast query Processing due to unnecessary column reads if only a subset of columns in a table are needed in a query

2.2 Symmetric Dynamic Programming Stereo Using Block Matching Guidance:

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

Explanation: In this paper, three stereo matching algorithms are investigated: Block Matching Stereo (BMS) represents local area matching techniques, Symmetric Dynamic Programming Stereo (SDPS) represents semi-global matching, and Graph Cuts Stereo (GCS) represents global matching.

Advantage: In this paper, three stereo matching algorithms are evaluated, which are representatives of well-known techniques of the field. Both local and semi-global methods are relatively fast and feasible for parallel implementation

Disadvantage:PECK schemes mostly depend on pairings and authenticated channel to achieve searchable encryption synopsis.

Disadvantage: This article proposes a technique which uses of less complex stereo matching algorithm.

2.3 Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data

Explanation: Cloud computing economically enables the paradigm of data service outsourcing. However, to protect data privacy, sensitive cloud data have to be encrypted before outsourced to the commercial public cloud, which makes effective data utilization service a very challenging task.

Advantage: We motivate and solve the problem of supporting efficient ranked keyword search for achieving effective utilization of remotely stored encrypted data in Cloud Computing

Disadvantage: Network traffic, which is absolutely undesirable in today's pay-as-you-use cloud paradigm. In short, lacking of effective mechanisms to ensure the file retrieval accuracy is a significant drawback.

2.4 Generalized Key Delegation for Wild carded Identity-Based and Inner-Product Encryption:

Explanation: Inspired by the fact that many e-mail addresses correspond to groups of users introduced the notion of identity-based encryption with wildcards (WIBE), which allows a sender to simultaneously encrypt messages to a group of users matching a certain pattern

Advantage: In addition to presenting two schemes achieving full security for polynomially many levels, we were also able to preserve the anonymity of the recipient in one of the schemes.

Disadvantage: Unfortunately, like in standard HIBE schemes, the hierarchical key derivation of a WIBE scheme has its limitations. In particular, it does not allow any deviation from the hierarchical.

2.5 An Efficient Public Key Encryption with Conjunctive Keyword Search Scheme Based On Pairings

Explanation: The Public Key Encryption with Conjunctive Keyword Search (PECK) scheme enables one to search a document included multiple encrypted keywords without compromising any original data information.

Advantage:A new secure and efficient SCF-PECK scheme based on pairings. Our proposed SCF-PECK scheme requires no pairing operations in PECK and Trapdoor phases

Disadvantage:PECK schemes mostly depend on pairings and authenticated channel to achieve searchable encryption synopsis.

3. EXISTING FUNCTIONS & ALGORITHMS:

Searchable encryption schemes enable the clients to store the encrypted data to the cloud and execute keywordsearch over cipher text domain. Due to different cryptography primitives, searchable encryption schemes can be constructed using public key based cryptography Multi-keyword Boolean search allows the users to input multiple query keywords torequest 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 supportboth conjunctive and disjunctive search. Christo Ananth et al. [6] proposed a system in which the complex parallelism technique is used to involve the processing of Substitution Byte, Shift Row, Mix Column and Add Round Key. Using S- Box complex parallelism, the original text is converted into cipher text. From that, we have achieved a 96% energy efficiency in Complex Parallelism Encryption technique and recovering the delay 232 ns. The complex parallelism that merge with parallel mix column and the one task one processor techniques are used. In future, Complex Parallelism single loop technique is used for recovering the original message.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 *etal.*'s search algorithm achieves better-than-linear search efficiency but results in precision loss.

3. PROBLEM FORMULATION

3.1 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 $F = \{f_1; f_2; :::; f_n\}$. Eachdocument f in the collection can be considered as asequence 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 the whole document collection F.

• I – The searchable encrypted tree index generated from$T$.

• 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.

• I_u– The encrypted form of D_u.

Vector Space Model and Relevance Score Function. Vector space model along with TF×IDF rule is widelyused 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 ofdocuments containing the keyword. In the vector spacemodel, 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 D_u and the IDF vector Qcan be calculated to quantify the relevance between the query and corresponding document. Following are the notations used in our relevance evaluation function:

• $N_{f;w_i}$– The number of keyword $w_i$in document f.

• N – The total number of documents.

• N_{w_i}– The number of documents that contain keyword w_i.

• $TF'_{f;w_i}$– The TF value of $w_i$in document f.

• IDF'_{w_i}– The IDF value of $w_i$in document collection.

• $TF_{u;w_i}$– The normalized TF value of keyword $w_i$stored in index vector D_u.

• IDF_{w_i}– The normalized IDF value of keyword $w_i$in document collection.The relevance evaluation function is defined as:

$$\text{RScore}(D_u;Q) = D_u \cdot Q = \Sigma_{w_i \in W_q}(TF_{u;w_i} \times IDF_{w_i}) \quad (1)$$

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

$$TF_{u;w_i}=TF'_{f;w_i}/(\sqrt{\Sigma_{w_i \in W}(TF'_{f;w_i})^2}; \quad (2)$$

where$TF'_{f;w_i}$= 1+ln$N_{f;w_i}$. And in the search vector Q, $IDF_{w_i}$is calculated as:

$$IDF_{w_i}=IDF'_{w_i}\sqrt{\Sigma_{w_i \in W_q}(IDF'_{w_i})^2}; \quad (3)$$

whereIDF'_{w_i}= ln(1 + N/N_{w_i}).

Keyword Balanced Binary Tree. The balanced binarytree is widely used to deal with optimization problems. 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 $D_u$for simplicity. Formally, the node u in our KBB tree is defined as follows:

$u = \langle ID;D; P_l; P_r; FID\rangle$.

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 $F = \{f_1; f_2; :::; f_n\}$ 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.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

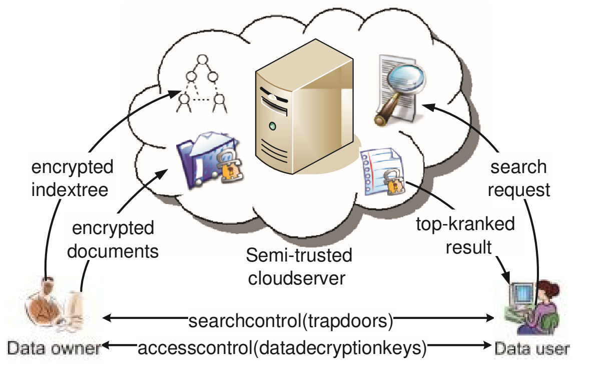
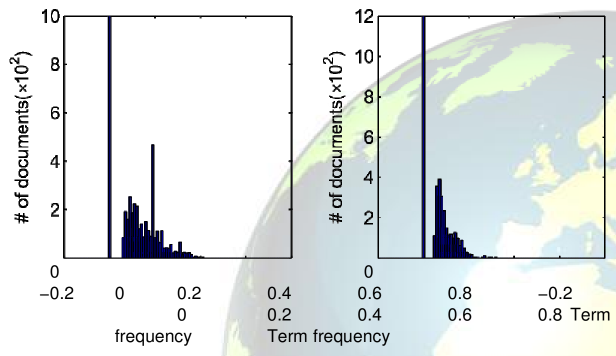


Fig. 1. The architecture of ranked search over encrypted cloud data



(a) (b)

(a) (b)

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

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) 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.

3.3 Design Goals

To enable secure, efficient, accurate and dynamic multikeyword ranked search over outsourced encrypted cloud

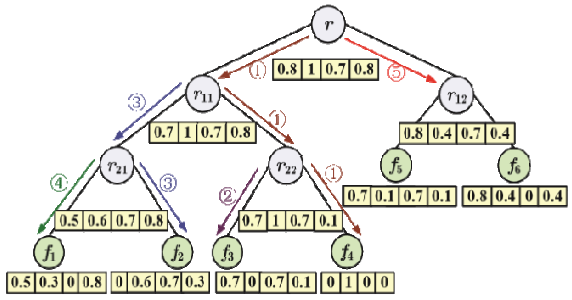


Fig. 3. An example of the tree-based index with the document collection.

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.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.

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.

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.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

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 $f_{FID}$to the query, which is calculated according to Formula (1). The *RList*stores the 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.

Algorithm 1 BuildIndexTree(F)

Input: the document collection F = $\{f_1,f_2,...,f_n\}$ with the identifiers FID = $\{FID|FID = 1,2,...,n\}$.

Output: the index tree T

```
1: for each document $f_{FID}$in F do
2:   Construct a leaf node $u$ for $f_{FID}$, with $u.ID$ = GenID(), $u.P_l = u.P_r = null$, $u.FID = FID$, and
     $D[i] = TF_{f_{FID},w_i}$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(), $u.P_l = u'$, $u.P_r = u''$, $u.FID = 0$ 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 $CurrentNodeSet$do
13:      Generate a parent node $u$ for $u'$and $u''$;
14:      Insert $u$ to $TempNodeSet$;
15:    end for
16:    Create a parent node $u_1$ for the $(2h - 1)$-th and $2h$-th node, and then create a parent node $u$ for $u_1$ 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$(D_u,Q)$ >$k^{th}score$then
3:         GDFS($u.hchild$);
4:         GDFS($u.lchild$);
5:     else
6:         return
7:     end if
8: else
9:     if RScore$(D_u,Q)$ >$k^{th}score$then
10:        Delete the element with the smallest relevance score from $RList$;
11:        Insert a new element ⟨RScore$(D_u,Q)$,$u.FID$⟩ and sort all the elements of $RList$;
12:    end if
13:    return
14: end if
```

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. The BDMRS scheme is designed to achieve the goal of privacypreserving in the known ciphertext 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 $I_u$and TD is exactly equal to that from D_u 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

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

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.

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.

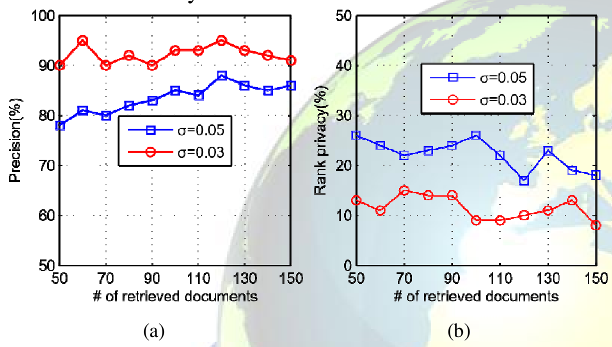


Fig. 4. The precision (a) and rank privacy (b) of searches with different standard deviation σ.

TABLE 2 :Precision test of basic scheme.

NO	Precision	NO	Precision
1	88%	9	96%
2	94%	10	86.7%
3	97%	11	87.5%
4	100%	12	100%
5	85%	13	82.3%
6	89%	14	100%
7	89%	15	100%
8	96%	16	71.1%

5. PERFORMANCE ANALYSIS

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). 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 an Intel 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]: $P_k = k'/k$, 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

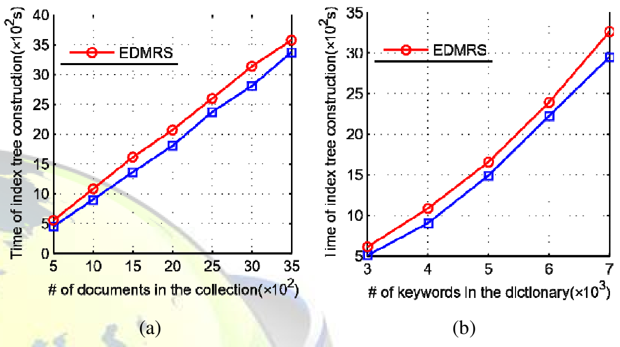


Fig. 5. Time cost for index tree construction: (a) for the different sizes of document collection with the fixed dictionary, m = 4000, and (b) for the different sizes of dictionary with the fixed document collection, n = 1000.

TABLE 3: Storage consumption of index tree.

Size of dictionary	1000	2000	3000	4000	5000
BDMRS (MB)	73	146	220	293	367
EDMRS (MB)	95	168	241	315	388

Variable$\sum \varepsilon_v$, the EDMRS scheme is supposed to obtain higher precision, and vice versa.

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 $(m \times m)$ 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 $(m \times m)$ matrix takes $O(m^2)$ time. As a whole, the time complexity for index tree construction is $O(nm^2)$.

5.2.2 Trapdoor Generation

The generation of a trapdoor incurs a vector splitting operation and two multiplications of a $(m \times m)$ matrix, thus the time complexity is $O(m^2)$, as shown in Fig. 6(a). Typical search requests usually consist of just a few keywords. Fig.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

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, the time cost of EDMRS scheme is a little higher than that of BDMRS scheme.

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.

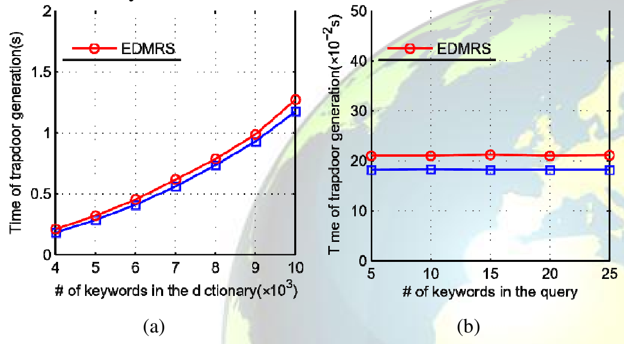


Fig. 6. Time cost for trapdoor generation: (a) for different sizes of dictionary with the fixed number of query keywords, t = 10, and (b) for different numbers of query keywords with the fixed dictionary, m = 4000.

5.2.4 Update Efficiency:

In order to update a leaf node, the data owner needs to update $\log n$nodes. Since it involves an encryption operation for index vector at each node, which takes $O(m^2)$ time, the time complexity of update operation is thus $O(m^2 \log n)$. We illustrate the time cost for the

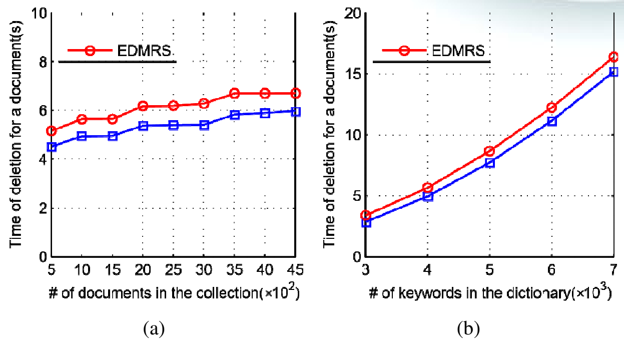


Fig. 7. Time cost for deletion of a document: (a) for the different sizes of document collection with the same dictionary, m = 4000, and (b) for the same document collection with different sizes of dictionary, n = 1000.

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.

7. 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 CryptologyEurocrypt 2004*. Springer, 2004, pp. 506–522.
[6] Christo Ananth, H.Anusuya Baby, "Encryption and Decryption in Complex Parallelism", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 3, Issue 3, March 2014,pp 790-795
[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, "Searchablesymmetric encryption: improved definitions and efficient constructions,"in*Proceedings of the 13th ACM conference on Computerand 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.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 19, April 2016

[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.
[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving Multikeyword fuzzy search over encrypted data in the cloud," in *IEEEINFOCOM*, 2014.
[15] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and NetworkSecurity*. Springer, 2004, pp. 31–45.
[16] Y. H. Hwang and P. J. Lee, "Public key encryption with Conjunctive keyword search and its extension to a multi-user system,"in*Proceedings of the First international conference on Pairing-BasedCryptography*. Springer-Verlag, 2007, pp. 2–22.
[17] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient Conjunctive keyword searches over encrypted data," in *Proceedings ofthe 7th international conference on Information and CommunicationsSecurity*. Springer-Verlag, 2005, pp. 414–426.
[18] D. Boneh and B. Waters, "Conjunctive, subset, and range Querieson encrypted data," in *Proceedings of the 4th conference on Theoryof cryptography*. Springer-Verlag, 2007, pp. 535–554.
[19] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.
[20] J. Katz, A. Sahai, and B. Waters, "Predicate encryption Supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology–EUROCRYPT 2008*. Springer, 2008, pp.146–162.
[21] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proceedings of the 6th Theory of Cryptography Conferenceon Theory of Cryptography*. Springer-Verlag, 2009, pp. 457–473.
[22] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in *Proceedings of the29th Annual international conference on Theory and Applications ofCryptographic Techniques*. Springer-Verlag 2010, pp. 62–91.
[23] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. L. Varna, S. He, M.Wu, and D.W. Oard, "Confidentiality-preserving rank-ordered search," in *Proceedings of the 2007 ACM workshop on Storage securityand survivability*. ACM, 2007, pp. 7–12.
[24] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+ r: Topkretrieval from a confidential index," in *Proceedings of the 12th International Conference on Extending Database Technology: Advancesin Database Technology*. ACM, 2009, pp. 439–449.
[25] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8,pp. 1467–1479, 2012.
[26] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in IEEE INFOCOM, April 2011, pp. 829–837.
[27] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in Proceedings of the 8th ACMSIGSAC symposium on Information, computer and communicationssecurity. ACM, 2013, pp. 71–82.