International Journal of Engineering & Technology, 7 (4.36) (2018) 736-743

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

EMSRSE: Efficient Multi-Keyword Synonym Based Ranked Search Technique Over Outsourced Encrypted Cloud Data

Veerraju Gampala[1*], Sreelatha Malempati[2]

[1]Department of Computer Science and Engineering, GMR Institute of Technology, Rajam.
Research Scholar, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India.
[2]Dept. of Computer Science and Engineering, R.V.R & J.C. College of Engineering, Chowdavaram, Guntur, Andhra Pradesh, India.
E-mail:lathamoturi@rediffmail.com
**Corresponding author E-mail:veerraju.g@gmrit.org*

Abstract

Recently, searching over encrypted cloud-data outsourcing has attracted the current researcher. Using cloud computing (CC), individuals and organizations are motivated to outsource their private and sensitive data onto the cloud service provider (CSP) due to less maintenance cost, great flexibility, and ease of access. However, the data should be encrypted using encryption techniques such as DES and AES before uploading to the CSP in order to provide data privacy and protection, which obsolete plaintext searching techniques over encrypted cloud data. Thus, this article proposes an efficient multi-keyword synonym-based ranked searching technique over encrypted cloud data (EMSRSE), which supports dynamic insertion and deletion of documents. The main objectives of EMSRSE are 1. To build an index search tree in order to store encrypted index vectors of documents and 2. To achieve better searching efficiency, a searching technique over the encrypted index tree is proposed. An extensive research and empirical result analysis show that the proposed EMSRSE scheme achieves better efficiency in comparison with other existing methods.

Keywords: *Cloud computing, multi-keyword search, synonym-based search, index search tree, ranked search.*

1. Introduction

In the past, CC has gained more attention from industry as well as academic people. As it becomes a new prototypical model for IT organizations, it can unify large resource computing, applications, storage, and on-demand network admittance to a collective pool of reckoning resources that achieve better efficiency, great flexibility, ease of access, and nominal maintenance cost [1]. With the attracted features of CC [2], more and more individuals and industry people are driven to upload their data such as financial records, health records, photos, albums, and emails onto the CSP instead of procuring own hardware and software to maintain the data in the local systems. As CSP cannot be trusted in terms of data outsourcing, the data encryption is recommended to provide data privacy, before it is uploaded to CSP. Since the data encryption does not have the answers for consumer queries on cloud data storage, traditional penetrating techniques cannot be used over encrypted searchable data in ciphertext domain. The existing searching techniques such as Google and Yahoo search over plaintext cannot be used directly over encrypted cloud data.
To overcome this issue, the entire encrypted data can be downloaded to the data consumer's local system and then decrypt. However, it is infeasible to apply because of high bandwidth utilization and computational overhead. Moreover, the data consumers may be interested to download only a few documents but not the entire document pool.
To address above issues, researchers are proposed several approaches to enable searching over encrypted cloud data ([3], [4], [5]), Such as fuzzy, single, similarity, and multi-keyword search. Among them, few approaches are practical applicability. These schemes mainly support the exact keyword search. Assume, a user searches a keyword "computer". He/ She will get zero results even the documents containing keywords like "system" or "laptop". Though they are similar kindin the computer field. The authors in [6] present a model for a protected multi-keyword pursuit over cloud information encryption. Moreover, it supports dynamic update operations such as insertion and deletion. To generate the index and the query vector, the vector space model and Term Frequency-Inverse Document Frequency (TFIDF) model are integrated. To provide efficient multi-keyword ranked pursuit, they built a special index tree structure named keyword balanced binary tree and proposed Greedy depth-first search algorithm.
As a result, this article dealing with how to design an efficient searchable encryption technique to support both the synonym-based search and multi-keyword ranked search in order to solve the above addressing issues. To address the issue distinctly, this article presents an efficient multi-keyword synonym-based ranked pursuit technique over encrypted cloud data. The proposed EMSRSE scheme uses WordNet [7] to generate a synonym set that enables a feature of the keyword dictionary. In addition, this scheme adapts a technique [8] to encrypt index vectors of documents and query vector in order to calculate the score of each document through encrypted vectors.
The major contributions of this article are as follows:

1. With the help of WordNet, a synonym set for a synonym-based search is generated in addition to multi-keyword search.
2. To achieve better searching efficiency in a multi-core processor system, a balanced searchable index tree and searching algorithm are proposed.

The rest of this article is prepared as follows: literature review is discussed in section 2, Problem formation is presented in section 3, which discuss the system model, threat model, and

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

notations. The EMSRSE scheme design is discussed in Section 4, Results, performance evaluation and analysis are explored in section 5. And finally concluding remarks are elaborated in section 6.

2. Literature survey

Various searching techniques have been proposed over encrypted cloud data. S. Deshpande [9] suggests a technique searching over encrypted cloud data using fuzzy keywords. They are used Edit distance to quantify keyword similarity and developed two techniques for building fuzzy keyword sets in order to attain improved storage and depiction overheads. Cong Wang et al. [10] proposes a method named ranked keyword pursuit over encrypted cloud data using techniques such as keyword frequency and order-preserving encryption. It supports only a single keyword at a time. Moreover, the keyword frequency decides the document files score. Rank is assigned to every file based on the relevance score of the corresponding file. Finally, top-ranked files are sent to users instead of all files. To enrich the search functionality N. Cao et al. [11] have proposed a system, which supports conjunctive keywords search. It is a privacy-preserving multi-keyword ranked pursuit technique using symmetric encryption.

Various researchers are employed a technology is known as Searchable Encryption (SE) that enables the searching over encrypted cloud data. SE allows the data owner to outsource the encrypted data and its related index to CSP [3,5,12]. As a result, all legitimate users are authorized to launch a query-based keyword search over the ciphertext domain. Various searchable techniques have been proposed [3,12-15] forsalient features such as security, searching accuracy, and computational overhead. The authors in [16] propose a model for a single keyword semantic-based pursuit over cloud data encryption affirming similarity ranking. This method returns not only the exact keyword matched files, but also the files comprise semantically related to the query keyword. The creators in [4] recommend various leveled bunching technique to help search semantics and furthermore quick scrambled information search on huge information condition. They propose various leveled strategy bunches the record documents in view of the base importance edge and after that partitions subsequent groups into sub-groups until the point that the limitation on the most extreme size of the bunch is coming.

However, the existing techniques are not well suited for the multi-keyword ranked search (MKRS), sincetheyemphasize on the single or boolean-keyword search. Primarily use a privacy-preserving scalar-product encryption (SPE) technique [8] to achieve multi-keyword ranked pursuit over encrypted data in CC, which achieves privacy preservation. Later, the SPE technique has become a popular tool for SE especially to examine security and computation time complexity. Keyword secure search techniques [6,17] employ the SPE to offer flexible dynamic operation namely insertion and deletion [6] not only to enhance search efficiency but also to support the user personalization search [17]. The authors in [18] present two techniques to support multi-keyword ranked pursuit to attain more accurate pursuit results and the synonym-based pursuit to support synonym queries over encrypted cloud data. The improved semantic feature extraction scheme E-TFIDF proposes by incorporating features extraction technique TFIDF that expands the accuracy of pursuit results. Veerraju et al. [19] present a complete study of keyword searching on encrypted cloud data and discuss the comparison of various schemes in terms of security.

To accomplish high efficiency and better precision over encrypted cloud data as, like plain-text search, research has been done. Wang et al. [10] suggest a secure ranked keyword pursuit technique that finds ranked keyword search to consider the keyword score relevancy. Boldyreva et al. [20] propose an Order-Preserving Encryption (OPE) scheme to accomplish ranked results. However, this scheme does not support trapdoor unlinkability. Sun et al. [21] offer a multi-keyword search method that uses keyword score relevancy and multidimensional tree to attain an efficiency of query searching. Yu et al. [22] suggest a secure multi-keyword top-k retrieval method, to retrieve top-k documents from CSP that employs homomorphic encryption to encrypt index vector and query vector in order to confirm high security. Offer a privacy-preserving multi-keyword ranked system for multi-keyword ranked pursuit over encrypted cloud data (MRSE) that uses coordinate matching.

3. Problem formation

3.1. System model

In this article, the system model comprises three entities: 1. the data owner, 2. Data consumer, and 3. CSP as shown in Figure 1.

The data owner (DO) has a pool of documents with sensitive information to outsource to the CSP. DO creates a dictionary based on keywords mined from all m documents based on Term Frequency Inverted Document Frequency (TFIDF) [23] In addition, this scheme generates the synonyms for each keyword through WordNet [7] that creates a keyword-synonym dictionary using keywords and corresponding synonyms. Using Equ.(1), the index vector is created for each document based on the keyword-synonym dictionary with the help ofthe term frequency (TF) weight of the keyword. To improve searching efficiency, a searchable balanced index tree is built for the document pool. To protect index tree privacy, data owner encrypts the index tree before uploading to CSP. In addition, all documents also encrypted using any standard encryption algorithm [24-25]. Afterward, the data owner uploads the encrypted document pool and index tree to the CSP. DO builds the trapdoor using query keywords of the legitimated data consumer and then sends decryption keys and trapdoor to him/her. Besides, the data owner also responsible to update the index tree based on insertion or deletion of documents.

Data consumer sends interested search keywords to DO and receives trapdoor from him. He/she sends trapdoor to the CSP and receives top k ranked encrypted documents from it. By then, He/she decrypts the documents using decryption keys.

The CSP stores the encrypted document pool and index tree of DO. Upon receiving the trapdoor from the data consumer, the CSP searches trapdoor over an encrypted index tree to obtain the top score ranked documents, in turn, returns top k ranked encrypted documents to the data consumer. Besides, the CSP also responsible to update the index tree and a document pool based on update information received from the DO.

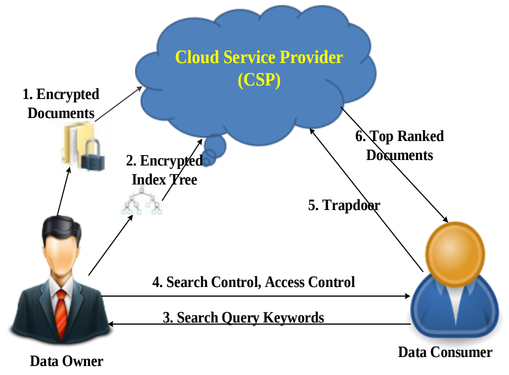


Figure 1: System architecture

3.2. Notations

D_P: Document pool, denoted as a pool of m documents $D_P = (D_1, D_2, \ldots, D_m)$.

E_P: Encrypted document pool stored in the CSP, denoted as a collection of m documents $E_P = (C_1, C_2, \ldots, C_m)$.

D_{ID}: The identity pool of encrypted documents E_P denoted as $D_{ID} = (D_{ID1}, D_{ID2}, \ldots, D_{IDm})$.
F: The unencrypted form of index vectors for D_P, denoted as a collection of m $F = (F_1, F_2, \ldots, F_m)$.
I: The encrypted form of index vectors for F, denoted as a collection of m $I = (I_1, I_2, \ldots, I_m)$.
D: Keyword-synonym dictionary, it contains n keywords and t synonyms of each keyword, denoted as

$$D_{n\times t} = (w_{11}, w_{12}, \ldots, w_{1t} \\ w_{21}, w_{22}, \ldots, w_{2t} \cdots \\ w_{n1}, w_{n2}, \ldots\ w_{nt})$$

$\mathcal{T}'$: The encrypted form of balanced index tree stored in CSP, it is built using I.
W: The keywords in the query entered by a data consumer.
Q: The query vector for keyword collection W.
TW: The encrypted form of Q, named as a trapdoor.
T_F: The sum of keyword frequency and corresponding synonyms (in keyword-synonym dictionary) frequency in the document.
L_k: It is a list to store retrieved top K document files in descending order according to the relevance score.
A_K: Least score in the L_k

3.3. Preliminaries

The TFIDF [23] model is employed in order to retrieve ranked search results, which are used in searchable encryption schemes. In this article, term frequency (T_F) is the sum of keyword frequency and corresponding synonyms frequency within a document. The inverse document frequency (IDF) shows the prominence of the term in the entire document pool. The relevance score of a keyword to a document is expressed as follows [28]:

$$Score(w_i, D_J) = \frac{1}{|D_J|}(1 + \ln f_{J,w_i}) \ln(1 + \frac{m}{f_{w_i}}) \quad \text{---} \qquad (1)$$

Where f_{J,w_i} means the term frequency of a keyword w_i in the document D_J, f_{w_i} Denotes the number of documents having the keywordw_i, $m$denotes the number of documents and $|D_J|$ denotes the number of indexed keywords.
Each document in the document pool is represented by the vector know as index vector, whose elemental values are calculated using Equ.(1). Moreover, the vector called the query vector that has IDF values of search query keywords represents the query. In the article [8], the authors suggest a secure k-nearest neighbor method to encrypt index and query vectors. It can encrypt two vectors and calculates the score between them. Firstly, the secret key (S, M_1, M_2) is created. Where, S is a bit vector used to split the index and query vectors into two random vectors each. The objective of the random vector generation is to ensure keyword anonymity in the plaintext vector. To encrypt the split vectors, two invertible matrices (M_1 and M_2)are utilized. The computation and security of this encryption method can be cited to[8].

3.4.EMSRSE Design

The EMSRSE scheme compromises of four phases namely, 1. Setup, 2. GenerateIndexTree, 3. GenerateTrapdoor, and 4. Search. The detailed description is as follows:
Setup: DO generates the secret key S_K={M_1, M_2, S}to encrypt the index and query vectors, where M_1 and M_2 are two invertible matrices and S is a bit vector.
GenerateIndexTree(D_P, S_K): Using BuildBalancedIndexTree(m, D_P) algorithm 1, DO bulids the unencrypted index tree based on index vectors of D_P. By then, each index vector F_u at node u is encrypted using S_K. Initially, F_u is split into F_u' and F_u''using S bit vector. If S[i]=0 then, F_u' and F_u'' are set as same as F_u[i]. If S[i]=1 then, F_u' and F_u'' are set to two random values whose sum equivalent to F_u[i]. Later, F_u' and F_u'' are encrypted using M_1 and M_2 as I_u={$M_1^T . F_u'$, $M_2^T . F_u''$}. Using the same process, all the index vectors within the tree are encrypted in order to generate an encrypted index tree.

3.5. Algorithm 1: Build Balanced Index Tree(m, D_P)

```
Input: m &$D_P$
Output: root node
Begin
1.  For each document $D_i$ in $D_P$ do
    Create a leaf node L for document$D_d$,    $L.ID = GenID()$,
        $L.child[i] = null for\ i = 1\ to\ branch\ degree\ b$,
        $L.did = did, F_d[j] = Score(w_j, D_d)\ for\ j = 1\ to\ dictionary\ size\ n$;
    Insert L into CurrentNodePool;
2.  End for;
3.  While (|CurrentNodePool|>1) do
      For each b nodes $u_1, u_2, .., u_b$ in CurrentNodePool
        Generate a parent node V such that   $V.ID = GenID(), V.child[j] = u_j\ for\ j = 1\ to\ b\ nodes V.did = null, and$
            D[i] = max{$u_i$.F[j] for i=1 to b} for each j=1 to n;
    /* D is index vector of node V */
    End for;
            Insert V to TempNodePool;
      End for;
            Generate a parent node V with the remaining nodes (<b) in CurrentNodePool like above;
                Insert V to TempNodePool;
                CurrentNodePool= TempNodePool;
                TempNodePool=0;
4.  End while;
5.          Return node left in CurrentNodePool as root;
6.  End;
```

GenerateTrapdoor(W): Based on interested keywords of the data consumer, the query vector Q is generated using D. If keyword available within D then, IDF value of the keyword is set to the corresponding dimension of the Q, otherwise set to zero. Q is split into $Q'$and Q'' using S. The splitting process is as same as index vector splitting but in reverse. Finally, the $Q'$and $Q''$are encrypted using M_1 and M_2 to generate trapdoor as TW= {$M_1^{-1} . Q'$, $M_2^{-1}.Q''$}.
Search(TW, $\mathcal{T}'$): Using TopK_search(u,l) algorithm 2, the CSP searches the trapdoor TW over encrypted index vector $\mathcal{T}'$ in order to generate top K ranked documents. At each node u of the $\mathcal{T}'$, the relevance score between the encrypted index vector and trapdoor is calculated using the inner product as follows:

$$\begin{aligned} \text{Relevance}(I_u, TW) &= \{M_1^T . F_u', M_2^T . F_u''\} \times \{M_1^{-1} . Q', M_2^{-1}.Q''\} \\ &= \{M_1^T . F_u' . M_1^{-1} . Q'\} + \{M_2^T . F_u'' . M_2^{-1}.Q''\} \\ &= \{F_u'^T . M_1 \times M_1^{-1} . Q'\} + \{F_u''^T . M_2 \times M_2^{-1}.Q''\} \\ &= F_u'^T . Q' + F_u''^T . Q'' \\ &= F_u^T . Q \\ &= \text{RelevanceScore}(F_u, Q) \end{aligned}$$

The search process is illustrated in Figure 2 using query vector Q = (0.91, 0, 0.8, 0.45) and K=3. It returns top 3 documents as D_{15}, D_7, and D_8. The red cross mark indicates that the search process stops at L_{21}.
Note: The relevance score between the encrypted index vector and trapdoor is as same as between unencrypted index vector and query vector.

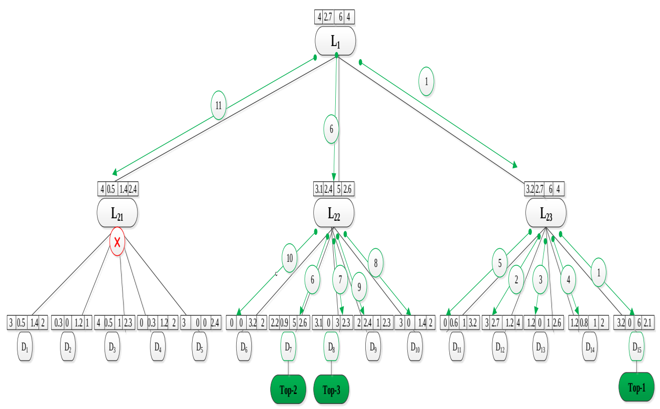


Figure 2: An example of the balanced index tree with the search process

3.6. Algorithm 2: TopK_search(u, l)

```
Input: node u & level l
Output: top-ranked list $L_k$
Begin
1.  If (u != leaf) then
            If (RelevanceScore ($F_u$, Q) > $A_k$) then
          Compute the scores of children and then sort in descending order
              For i=1 to children of u do
                  TopK_search(u.child[i], l+1)
              End for;
          Else
              Return;
            End if;
2.  Else
      If($|L_k|$ < K) then
        Insert document $F_d$ into $L_k$ according to Score ($F_u$, Q);
      Else
          If (RelevanceScore ($F_u$, Q) > $A_k$ ) then
              Delete $A_k$ from RankedList $L_k$;
              Insert document $F_d$ into $L_k$ according to RelevanceScore ($F_u$, Q);
          Else
            Return;
          End if;
      End if;
    Sort $L_k$ in descending order;
3.  End if;
4.  Return $L_k$;
End;
```

3.7. Balanced index tree update

The balanced index tree needs to update after the insertion or deletion of a document. The update operation is based on the identity of a document, but not required to access document data. The update process is as follows:

$\{\mathcal{T}'_s, C_{id}\}$ = GenTreeUpdateInfo $(S_K, \mathcal{T}_s, id, UpdateType)$: This procedure returns tree-update information, $\{\mathcal{T}'_s, C_{id}\}$, in turn, send to the CSP. Here, $\mathcal{T}'_s$ is updated sub-tree and C_{id} is an encrypted document. The parameter UpdateType ϵ {insert, delete} indicates either insertion or deletion of a document D_{id} in the index tree. $\mathcal{T}_s$ indicates a set of nodes to update. In order to ease the communication cost, the data owner stores a copy of the unencrypted index tree. For example, if you wish to delete the document D_{12} from the index tree shown in Figure 2, then UpdateType equals to delete and the sub-tree $\mathcal{T}_s$ contains a set of nodes $\{L_{22}, L_1\}$. The DO deletes the leaf node, which contains document identity 12 and index vector and then updates the index vectors of the nodes L_{22} and L_1 in order to update $\mathcal{T}_s$. While deletion, the index tree may be unbalanced. To avoid this, DO replaces the deleted node with a dummy node. Moreover, its identity is set to null and index vector elemental values are set to zero. Finally, the data owner encrypts the index vectors in $\mathcal{T}_s$ using S_K in order to generate an encrypted sub-index tree $\mathcal{T}'_s$ and C_{id} set to null. If UpdateType is equal to insert, then the DO work is to create a leaf node with the document identity id for the new document D_{id}. Using keyword-synonym dictionary, DO generates index vector for new document and inserts this new leaf node into $\mathcal{T}_s$. However, it also updates vectors of other nodes in the sub-tree $\mathcal{T}_s$. Preferably, DO replaces the dummy nodes with new leaf nodes. Finally, sub-tree $\mathcal{T}_s$ encrypted using S_K in order to generate $\mathcal{T}'_s$ and the document D_{id} is encrypted in order to generate C_{id}.

$\{\mathcal{T}', C'\}$ = UpdateIndexTree ($\mathcal{T}$, C, UpdateType, $\mathcal{T}'_s, C_{id}$): CSP runs this procedure to replace sub-tree $\mathcal{T}_s$ (encrypted sub-tree in the $\mathcal{T}$) with $\mathcal{T}'_s$ in order to generate $\mathcal{T}'$. If the update operation is insertion, then the CSP inserts the encrypted document C_{id} into C to generate new C'. If the update operation is deletion, then CSP deletes the C_{id} from C to generate new C'.

4. Results and discussions

This section presents the result analysis of the proposed index tree that constructively proposes a search algorithm over the encrypted search tree. The experiments are conducted for existing scheme BDMRS [6] and SMSRQE [28] in addition to the proposed EMSRSE. To compare and analyze the performance of the proposed scheme, schemes are implemented in Java language using Spring Tool Suite (STS) and tested on OpenStack instance with a flavor of 2.20 GHz Intel Core(TM) i5 processor and 8GB RAM. In addition, the secret key (M_1, M_2, S), and the dictionary are stored in a text file on the data owner system. To acquire more accurate and efficient analysis, all the tests are conducted on three datasets namely, National Science Foundation research awards [31], Internet Request for Comments (RFC) [29], and own created dataset. Moreover, the performance of the proposed method EMSRSE in comparison with other existing methods SMSRQE and BDMRS is estimated using efficiency.

4.1. Efficiency

Using time cost, the proposed scheme's efficiency is measured to generate an encrypted index tree, trapdoor, and searching over the encrypted tree.

4.2. Encrypted index tree building

The time required to construct an encrypted index tree incurs an index vector generation for d_p, unencrypted index tree construction, and finally building an encrypted index tree. To generate index vector for each document within d_p, dictionary keywords and synonyms are searching within the corresponding document, based on the availability of the term index vector is created. Using the proposed index tree-building algorithm, the unencrypted index tree is built with index vectors. Finally, the index vector at every node is encrypted using the secret key, which involves a vector splitting operation, two matrices transpose, and two multiplications of size (n × n) in order to generate an encrypted index tree. To compare dependency on dataset size (MB), the encrypted index tree is built for three different size datasets with a fixed number of keywords equal to 2000 as shown in Figure 3.

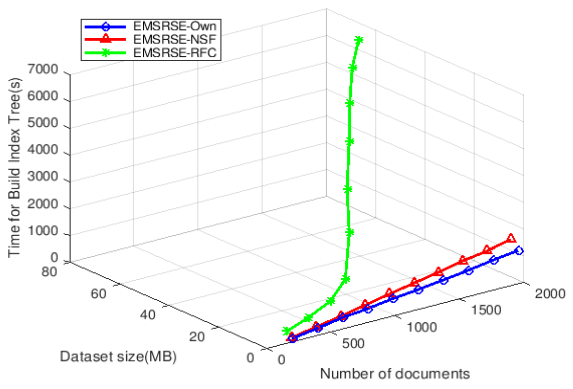


Figure 3: Time for index tree building for different size datasets and a fixed number of dictionary keywords |n × t| = 2000×3.

Searching dictionary terms within a large size document consume more time. As a result, EMSRSE-RFC consumes more time in comparison with EMSRSE-NSF and EMSRSE-Own due to the size of RFC dataset is more when compared to NSF and Own datasets, it is observed in Figure 3. Moreover, EMSRSE-NSF consumes more time in comparison with EMSRSE-Own because the size of the NSF dataset is more when compare to Own dataset. In addition, EMSRSE-NSF and EMSRSE-Own lines are linear with a number of documents in the dataset because dataset size and documents are increasing linearly, but the EMSRSE-RFC line is nonlinear. The dataset size and time cost are compared in Figure 3.

The time cost to build the index tree for a different size document pool with a fixed number of dictionary keywords |n × t| = 2000×3 for the proposed index tree in comparison with existing BDMRS [6] and SMSRQE [28] is shown in Figure 4(a).It shows that time cost to build index tree for proposed EMSRSE consumes less time when compared to existing BDMRS scheme due to the more number of internal nodes generates while building BDMRS index tree when compare to EMSRSE index tree. The SMSRQE consumes less time in comparison with proposed EMSRSE due to EMSRSE encrypts vector at internal nodes in addition to document vectors, whereas SMSRQE encrypts only document vectors but it is one-time work at data owner. Moreover, the time cost to construct index tree is linear with the number of documents, which is observed from the Figure 4(a).

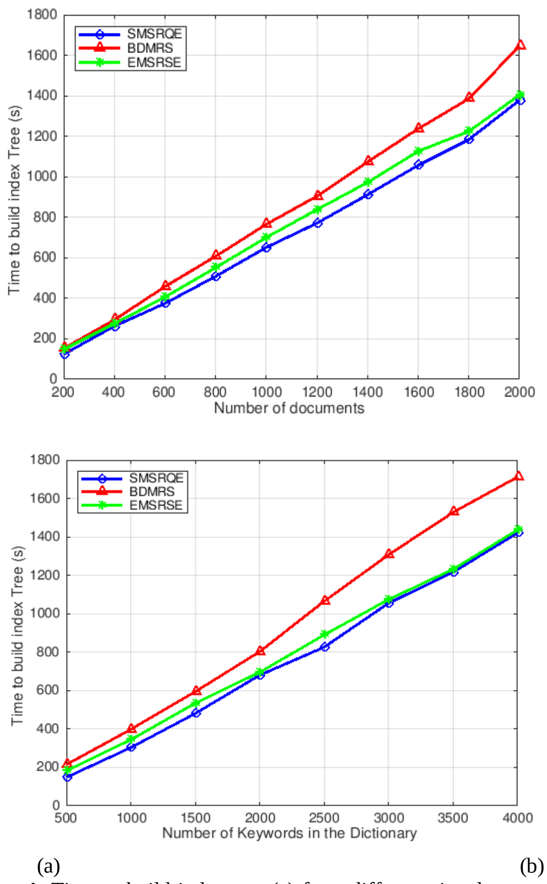


Figure 4: Time to build index tree (a) for a different size document pool with a fixed number of dictionary keywords |n × t| = 2000×3. (b) For the different sizes of dictionary keywords with fixed size document pool |m|= 1000.

The time cost to build index tree for different sizes of dictionary keywords and fixed size document pool |m| = 1000 for the proposed EMSRSE scheme in comparison with other existing methods BDMRS and SMSRQE is shown in Figure 4(b). In Figure 4(b), EMSRSE consumes less time in comparison with existing BDMRS scheme due to 2001 index vectors need to encrypt in the BDMRS scheme whereas only 1251 index vectors need to encrypt in the EMSRSE scheme in order to construct encrypted index tree. Moreover, the SMSRQE scheme consumes less time in comparison with EMSRSE due to only1000 index vectors need to encrypt in SMSRQE. In addition, no need to construct an index tree in the SMSRQE scheme. The Figure 4(b) shows that the time cost to build the index tree is almost proportional to the number of keywords in the dictionary.

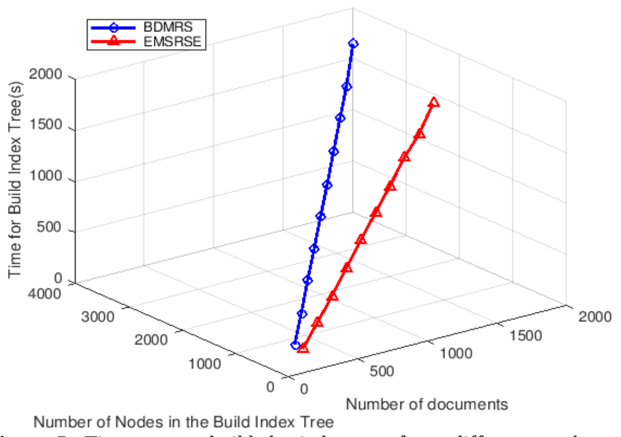


Figure 5: Time cost to build the index tree for a different number of documents and nodes with a fixed number of dictionary keywords |n × t| = 2000×3.

The Time cost to build the index tree for a different number of documents and index tree nodes with a fixed number of dictionary keywords |n × t| = 2000×3 for the proposed EMSRSE scheme in comparison with existing scheme BDMRS is shown Figure 5. From the Figure 5, the proposed EMSRSE consumes less time when compared to existing BDMRS scheme due to the less number of internal nodes are generated to build EMSRSE index tree when compare to BDMRS index tree. Moreover, the number of nodes required for each set of documents to generate index tree are compared in Table 1for both schemes. Figure 5 shows that the time cost to construct index tree is linear with the number of documents and number of nodes in the index tree.

Table 1: Time to Build an Index Tree for a Different Number of Documents and Nodes with a Fixed Number of Dictionary Keywords |n × t| = 2000×3

S.No	Number of Keywords fixed	Number of Documents	Number of nodes in the BDMRS Tree	Time for BDMRS(s)	Number of nodes in the EMSRSE Tree	Time for EMSRSE(s)
1	2000	100	202	84.9	125	76.2
2	2000	200	402	152.8	251	148
3	2000	300	603	220.4	376	220.7
4	2000	400	802	293.7	501	275.1
5	2000	500	1001	387.4	625	334.2
6	2000	600	1203	457.9	750	406.3
7	2000	700	1402	536.7	877	488.4
8	2000	800	1602	607.4	1002	551.5
9	2000	900	1804	688.1	1127	654
10	2000	1000	2001	765.5	1251	700.2
11	2000	1200	2403	904.3	1501	838.1
12	2000	1400	2802	1073.8	1752	972.3
13	2000	1500	3002	1156.9	1876	1050.2
14	2000	1600	3202	1237.1	2001	1126.3
15	2000	1800	3604	1385.8	2251	1224.7
16	2000	2000	4001	1646.1	2501	1404.1

4.3.Storage efficiency

The space complexity of the proposed balanced encrypted index tree depends on the number of nodes created for the index tree and dictionary size. The index tree nodes, in turn, depends on the number of documents. In an encrypted index tree, every node stores two vectors of dictionary size n. Thus, the space complexity of the proposed index tree is (2×n×o), where o is the number of nodes in the index tree, i.e. O (no). However, each vector element consumes eight bytes of storage due to a vector defined as double in Java language. As shown in Table 2(a), when the number of keywords is fixed (n=2000), the storage cost of the index tree is increasing with the number of documents. The Table 2(b) shows that the storage cost of index tree increasing with the number of keywords when fixed number of documents (m=1000). The Table 2 shows that the proposed EMSRSE index tree consumes less storage cost when compared to existing BDMRS index tree.

Table 2: Storage cost of Index tree

Number of Documents	Number of nodes in the BDMRS Tree	Storage cost of BDMRS(MB)	Number of nodes in the EMSRSE Tree	Storage cost of EMSRSE (MB)
500	1001	32	625	20
1000	2001	64	1251	40
1500	3002	96	1876	60
2000	4001	128	2501	80

(a)

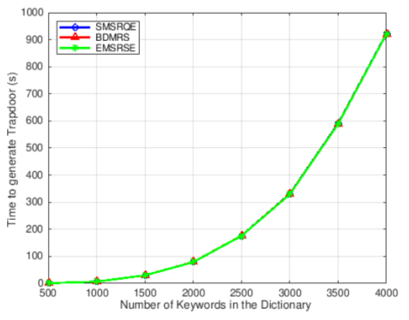
Number of Keywords	Storage cost of BDMRS(MB)	Storage cost of EMSRSE (MB)
500	16	10
1000	32	20
1500	48	30
2000	64	40
2500	80	50
3000	96	60
3500	112	70
4000	128	80

(b)

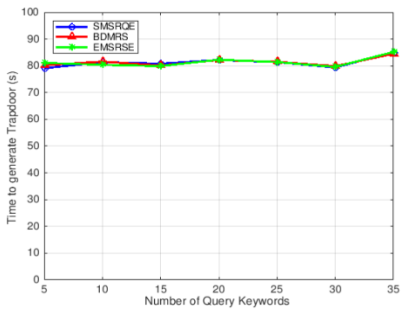
4.4. Trapdoor generation

The trapdoor generation incurs two matrices' (M_1 and M_2) inverse of size (n × n), query vector generation, query vector splitting operation, and two multiplications of size (n × n) matrix. The time complexity for matrix inverse is O (n^3) and matrix multiplication O (n^3). Thus, the overall time complexity for trapdoor generation is O (n^3) as shown in Figure 6(a). The Figure 6(a) shows that the graph is equivalent to $y=x^3$. The time required to generate trapdoor for a different number of dictionary keywords for proposed EMSRSE in comparison with other existing methods BDMRS and SMSRQE is shown in Figure 6(a).

The Figure 6(a) shows that the time cost for all the three schemes almost equal in order to generate trapdoor for a different number of dictionary keywords.



(a)



(b)

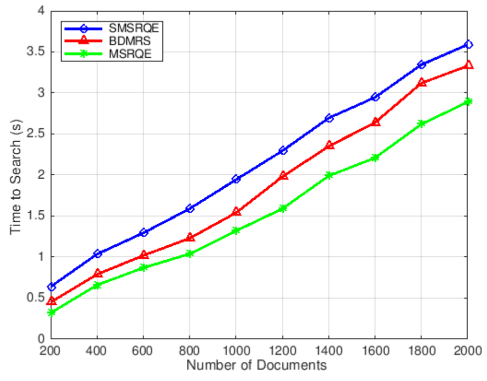
Figure 6: Time cost to generate trapdoor (a) for a different number of dictionary keywords. (b) For a different number of query keywords with a fixed size dictionary keywords |n| = 2000.

The time cost to generate trapdoor for a different number of query keywords with a fixed size dictionary keywords |n| = 2000 for all the three schemes are compared in Figure 6(b). The Figure 6(b) shows that the query keywords are not a significant influence in the trapdoor generation. Moreover, all the three schemes are consumed almost equal time to generate trapdoor for a different number of query keywords.

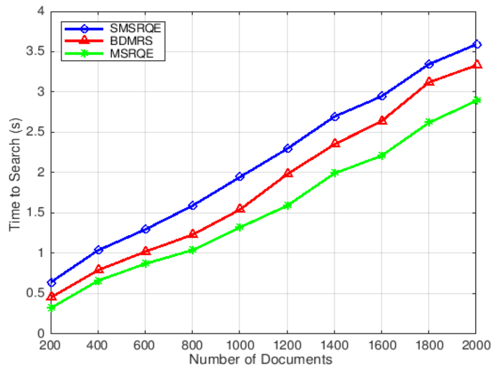
4.5. Search efficiency

During search over an encrypted index tree, if the score at node u is larger than the minimum score in the resultant ranked list L_k, the CSP examines the children of node u; else, it returns. As a result, a large number of nodes are not examined during the real search process.

The time cost to search over an encrypted index tree for a different number of documents with a fixed number of dictionary keywords |n| = 2000 for all the three schemes is compared in Figure 7(a). From the Figure 7(a), the proposed EMSRSE scheme consumes very less time when compared to existing BDMRS and SMSRQE due to the proposed balanced index tree and search algorithm over an encrypted index tree.



(a)



(b)

Figure 7: Time cost to search over an encrypted index tree (a) for a different number of documents with a fixed number of dictionary keywords |n| = 2000. (b) For a different number of retrieved documents with a fixed number of dictionary keywords |n| = 2000 and a fixed number of documents |m| = 1000.

The Figure 7(a)shows that the proposed EMSRSE consumes less time cost to search over an encrypted index tree in comparison with existing BDMRS due tothe less number of nodes in the index tree of EMSRSE scheme. Finally, it is observed that the proposed EMSRSE scheme more efficient in terms of search efficiency in comparison with other existing schemes such as BDMRS and SMSRQE.

The time cost to search over an encrypted index tree for a different number of retrieved documents with a fixed number of dictionary keywords |n| = 2000 and a fixed number of documents |m| = 1000 for all the three schemes is compared in Figure 7(b). The Figure 7(b) shows that the time cost to search over an encrypted index tree is not influenced by a number of documents retrieved for all the three schemes. Nevertheless, the time cost of EMSRSE is very less when compared to existing BDMRS and SMSRQE schemes. Thus, the proposed EMSRSE is more efficient than the BDMRS and SMSRQE schemes in terms of search efficiency.

5. Conclusion

In this article, an efficient multi-keyword synonym based ranked search technique is proposed, which supports dynamic insertion and deletion of documents. To acquire better search efficiency than a linear search, the balanced index tree is proposed and proposes a searching technique over an encrypted index tree. In addition, the parallel search over the index tree further reduces the search time cost. In the EMSRSE scheme, the search process has computing and ranking relevance scores of relevant documents rather than all documents in the index tree. Moreover, the extensive research and experimental results show that the proposed scheme achieves better search efficiency in comparison with other existing SMSRQE and BDMRS schemes. In the future, multiple DO system model will be explored. Search approaches over encrypted cloud data can be extended to support anaphora resolution and other natural language processing technology.

References

[1] Mell P & Grance T, *The NIST definition of cloud computing*, NIST special publication, (2011).

[2] Gampala V, Inuganti S & Muppidi S, “Data security in cloud computing with elliptic curve cryptography”, *International Journal of Soft Computing and Engineering (IJSCE)*, Vol.2, No.3, (2012), pp.138-141.

[3] Song DX, Wagner D & Perrig A, “Practical techniques for searches on encrypted data”, *IEEE Symposium on Security and Privacy*, (2000), pp.44-55.

[4] Chen C, Zhu X, Shen P, Hu J, Guo S, Tari Z & Zomaya AY, “An efficient privacy-preserving ranked keyword search method”, *IEEE Transactions on Parallel and Distributed Systems*, Vol.27, No.4, (2016), pp.951-963.

[5] Li H, Liu D, Dai Y, Luan TH & Shen XS, “Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage”, *IEEE Transactions on Emerging Topics in Computing*, Vol.3, No.1, (2015), pp.127-138.

[6] Xia Z, Wang X, Sun X & Wang Q, “A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data”, *IEEE Trans. Parallel Distrib. Syst.*, Vol.27, No.2, (2016), pp.340-352.

[7] Miller GA, “WordNet: a lexical database for English”, *Communications of the ACM*, Vol.38, No.11, (1995), pp.39-41.

[8] Wong WK, Cheung DWL, Kao B & Mamoulis, N, “Secure knn computation on encrypted databases”, *International Conference on ACM SIGMOD Management of data*, (2009), pp.139-152.

[9] Li J, Wang Q, Wang C, Cao N, Ren K & Lou W, “Fuzzy keyword search over encrypted data in cloud computing”, *Proceedings of IEEE Infocom*, (2010), pp.1-5.

[10] Wang C, Cao N, Li J, Ren K & Lou W, “Secure ranked keyword search over encrypted cloud data”, *IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, (2010), pp.253-262.

[11] Cao N, Wang C, Li M, Ren K & Lou W, “Privacy-preserving multi-keyword ranked search over encrypted cloud data”, *IEEE Transactions on parallel and distributed systems*, Vol.25, No.1, (2014), pp.222-233.

[12] Li R, Xu Z, Kang W, Yow KC & Xu CZ, “Efficient multi-keyword ranked query over encrypted data in cloud computing”, *Future Generation Computer Systems*, Vol.30, (2014), pp.179-190.

[13] Goh EJ, “Secure indexes”, *IACR Cryptology ePrint Archive*, (2003).

[14] Chang YC & Mitzenmacher M, “Privacy preserving keyword searches on remote encrypted data”, *International Conference on Applied Cryptography and Network Security*, (2005), pp. 442-455.

[15] Curtmola R, Garay J, Kamara S & Ostrovsky R, “Searchable symmetric encryption: improved definitions and efficient constructions”, *Journal of Computer Security*, Vol.19, No.5, (2011), pp.895-934.

[16] Xia Z, Zhu Y, Sun X & Chen L, “Secure semantic expansion based search over encrypted cloud data supporting similarity ranking”, *Journal of Cloud Computing*, Vol.3, No.1, (2014).

[17] Fu Z, Ren K, Shu J, Sun X & Huang F, “Enabling personalized search over encrypted outsourced data with efficiency improvement”, *IEEE transactions on parallel and distributed systems*, Vol.27, No.9, (2016), pp.2546-2559.

[18] Zhangjie F, Xingming S, Nigel L & Lu Z, “Achieving Effective Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Synonym Query”, *IEEE Trans. Consumer Electronics*, Vol.60, No.1, (2014), pp.164-172.

[19] Veerraju G & Sreelatha M, "A Study on Privacy Preserving Searching Approaches on Encrypted Data and Open Challenging Issues in Cloud Computing", *International Journal of Computer Science and Information Security*, Vol.14, No.12, (2016), pp.294-307.
[20] Boldyreva A, Chenette N, Lee Y & Oneill A, "Order-preserving symmetric encryption", *Advances in Cryptology-EUROCRYPT, Springer*, (2009), pp.224–241.
[21] Sun W, Wang B, Cao N, Li M, Lou W, Hou YT & Li H, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking", *8th ACM SIGSAC symposium on Information, computer and communications security*, (2013), pp.71-82.
[22] Yu J, Lu P, Zhu Y, Xue G & Li M. Towards secure multi-keyword top-k retrieval over encrypted cloud data. *IEEE Transactions on Dependable and Secure* Computing, Vol.10, No.4, (2013), pp.239–250.
[23] Schütze H, Manning, CD & Raghavan P. *Introduction to information retrieval*. Cambridge University Press, Vol.39, (2008).
[24] National Bureau of Standards, Data Encryption Standard, U.S. Department of Commerce, Washington, DC, USA, 1977.
[25] Dobbertin H & Rijmen V, *Advanced Encryption Standard-AES: 4th International Conference, AES 2004, Bonn, Germany, Revised Selected and Invited Papers*, Springer Science & Business Media, (2005).
[26] Zobel J & Moffat A, "Exploring the similarity space", *ACM SIGIR Forum*, Vol.32, No.1, (1998), pp.18–34.
[27] Veerraju G & Sreelatha M, "Secure Semantic Multi-keyword Synonym Ranked Query over Encrypted Cloud Data", *International Journal of Engineering and Technology*, Vol.8, No.1, (2016), pp. 98-107.
[28] Request for comments, (2014).
[29] National Science Foundation Research Awards Abstracts 1990-2003.