Secure Semantic Multi-keyword Synonym Ranked Query over Encrypted Cloud Data

Veerraju Gampala #1, Sreelatha Malempati #2
[1]Research Scholar Achrya Nagarjuna University, Guntur, Andhra Pradesh, India.
Email: veerraju.g@gmrit.org
[2]Department of Computer Science and Engineering,
R.V.R & J.C. College of Engineering,
Chowdavaram, Guntur, Andhra Pradesh, India.

Abstract-- **Cloud computing is a new auspicious technology and it greatly accelerates the development of large-scale data storage, computing and distribution. With the benefit of cloud computing, data owners are driven to outsource their penetrating and complex data from local machines to the public cloud due cost, efficiency and less hands on management. However, security and privacy become major security issues when data owners outsource their private data onto untrusted public cloud servers. Hence, to protect data, penetrating data has to be encrypted before outsource onto the cloud system. However, traditional keyword plain text search is obsolete. Hence, achieving encrypted cloud data search is utmost important. Existing searching techniques over encrypted cloud data considers only exact or fuzzy keyword or multi-keyword, but not semantic and synonym based ranked searching using multi-keyword. Hence, how to facilitate an effective searchable system to support semantic and synonym query is still challenging issue. This paper defines and proposes an effective solution to solve the problem of semantic multi-keyword synonym query over encrypted cloud data by retrieving top k scored documents, termed as SMSRQE. In this SMSRQE, keywords and corresponding synonyms are mapped to design multi-keyword synonym dictionary. Hence, it gives accurate search results for data user interested keywords or synonyms or both. To find semantically related documents, we used Latent Semantic Index (LSI). Our experiments show the SMSRQE methodology gives effective, efficient and accurate search results over the current solutions.**

Index terms-- cloud computing, multi-keyword search, synonym keyword search, ranked search, semantic query

I. INTRODUCTION

Cloud computing is the extensive fantasized dream of computing as a service, where consumers of cloud can remotely store their sensitive and private data like emails, health records and personal photos onto the cloud. With the quality applications, we can get services on-demand from a shared environment with computing resources from anywhere in the world. It is a great tractability and cost saving to motivating both industry and individual to outsource their local private complex data management system onto the cloud. The innovative cloud-computing paradigm has generated a substantial interest in both industry and academia, resulting in a number of famous marketable and individual cloud computing services provided by several cloud service providers, examples from Amazon web services, Google App engine, Microsoft Azure, Aneka, Yahoo, and Sales force. Top database vendors such as Oracle and IBM with IBM BlueMix are adding cloud support to their databases.

The main hazard of data confidentiality is origins in the cloud itself. Reports of privacy fissures and data loss in cloud computing system appear from time to time. When the data owners outsource their private penetrating data onto the cloud, the cloud facility providers are able to govern and observe the data and communication between cloud users and the cloud server. When an organization or institute deployed an application, it will give authorization to its different level staff to access different document files from cloud with different access permissions. Each document is a combination of a set of keywords, and staff as authorized consumers can access documents of their interests by applying query onto the cloud with some keywords. In such architecture, how to protect user privacy from the cloud, which is third party outside organization, hence the security boundary becomes a crucial problem. User privacy means user can outsource or access sensitive data from cloud without revealing his or her identity to anybody. User privacy classified into two types, search privacy and access privacy. Search privacy defined, as the cloud facility provider identifies nothing about what the user doing with searching keywords. Access privacy defined, as cloud identifies nothing about what the cloud facility provider returns as a result to the cloud user.

To protect data and achieve privacy preserving, data owners before outsourcing onto the cloud should encrypt data. However, traditional plain text searching techniques is obsolete. The solution to this is to download all encrypted files and decrypt in the user machine. But it takes more network bandwidth cost, moreover large amount of data, documents is outsourced onto the cloud, and to download all documents to local storage may not suffice. This is a very challenging issue; it degrades performance of system usability and scalability.

On the other hand, to retrieve data effectively, the cloud server has to calculate relevance-ranking documents as a result, instead of returning all documents as a result. Such ranked search enables consumers to find the most relevant documents quickly. Sort the matching content of documents according to its score. This searching technique eliminates unnecessary network traffic by sending only relevant documents, which is highly desirable in pay as you use cloud paradigm model. To improve the accuracy of the search results and enhanced searching, it is also necessary to support multiple keyword ranked search systems. We can observe today's web search engines (Google search, Yahoo search); in these engines, data users can provide a set of keywords to search instead of the single, to retrieve most relevant data. In coordinate matching the number of matches as possible as, is done with the efficient parallel measure among such multi-keyword enhance the result relevance, and has been used in plain text information retrieval system (IR).

In the literature survey, there exist searching techniques on encrypted data. Documents retrieved securely by placing query on encrypted data using single keyword and multi-keyword search. The drawback is users may not get accurate and relevant documents. Some recent searching techniques propose using Boolean algebra keyword search as an effort to improve the searching tractability; they are still not adequate to provide consumers with acceptable results with ranking functionality. How to design secure multi-keyword synonyms search over encrypted cloud data in cloud computing is still open challenging problem.

In this paper, we propose a method to solve the problem of Semantic Multi-keyword Synonym Ranked Query over encrypted cloud data (SMSRQE), while preserving strict system wise privacy in cloud computing paradigm. The main contributions are four aspects: the first one is semantic search based on latent semantic index, the second multi-keyword search, the third synonym based search and the last similarity ranked search. Our experiment analysis further shows efficient and accurate top k documents retrieval of the proposed method.

II. Related Work

Many searching techniques over encrypted cloud data have proposed. S. Deshpande [7] suggested a technique searching over encrypted cloud data using fuzzy keywords. They used edit distance to quantify keyword similarity and developed two techniques on constructing fuzzy keyword sets to achieve optimized storage and representation overheads. Cong wang et al. [1] Has proposed a method ranked keyword search over encrypted cloud data using keyword frequency and order preserving encryption. It supports only single keywords at a time. Is the keyword frequency deciding document file score. Rank given to every file based on the relevance score of that file. Top ranked files have sent to users instead all files. To enrich search functionality N. Cao et al. [2] Have proposed a scheme supporting conjunctive keywords search. It is privacy – preserving multi-keyword ranked search technique using symmetric encryption. M.Chuah et al. [6] Proposed a solution for fuzzy multi-keyword search over encrypted cloud data using privacy aware Bed Tree. They used a co-occurrence probability approach to identify useful multi-keywords for published data documents and relevant fuzzy keyword sets constructed using edit distance. They constructed index tree for all data documents, where each leaf node having the hash value of a keyword, one or two data vectors that represents n- gram of that keyword and bloom filters for each edit distance value. C. Wang et al. [3] Suggested a technique to build storage efficient similarity keyword set with a given document collection, edit distance as a similarity metric. Based on that, they built a private trie traverse-searching the index to achieve similarity search functionality with constant time complexity. C.Yang et al. [4] Designed a scheme adopting three sparse matrices instead dense matrix pair in MRSE to encrypt index, they combined their scheme with a bloom filter to gain the ability for index updating. D.X.Song. D et al. [8] Proposed a method, remote searching over encrypted data using an untrusted server and provided proof for the resulting crypto system. They supported controlled, hidden search and query isolation. C. Wang et al. [10] Proposed an efficient Ranked Searchable Symmetric Encryption scheme (RSSE). It supports ranked keyword search, security guarantee and efficiency with minimum communication cost. B. Wang et al. [11] suggested a novel multi-keyword fuzzy search scheme by exploiting the locality sensitive hashing techniques. They achieved fuzzy matching through algorithmic design rather than expanding index file and it eliminates the need of predefined dictionary. N. Cao et al. [12] proposed two Multi - keyword Ranked Search over Encrypted cloud data (MRSE) schemes based on a similarity measure coordinate matching while meeting different privacy requirements in two different threat models. They enhanced ranked search mechanism to support dynamic data operations. In no above techniques supported semantic search and synonym-based search.

In our work, we designed an efficient secure semantic multi-keyword synonym ranked searching technique over encrypted cloud data by employing Latent Semantic Indexing to search semantically similar documents.

III. SYSTEM MODEL

In this paper, we considered a cloud computing system model involves three different entities. Those are Data Owner, Cloud Service Provider and Data user as illustrated in Figure1. The responsibility of each entity is as follows:

Data Owner (DO): DO has a collection data documents DC= {d_1, d_2…, d_m} with sensitive information to be outsourced to the cloud server. To provide data privacy, the documents are encrypted before outsourcing. DO creates a dictionary based on keywords extracted from the all m documents based on Term Frequency Inverted Document Frequency (TFIDF) [13] which is described in section IV. The dictionary includes synonyms of each keyword from the thesaurus [14]. The Dictionary is having n keywords, and for each keyword may have t synonyms, so that the dictionary size is $n \times t$. DO creates an index vector for each document based on the keywords extracted from the document. The size of the index vector is equal to the number of keywords in the dictionary that is n. Each dimension in the index vector stores sum of the frequency of keyword and corresponding synonyms in the dictionary is denoted as term frequency (TF) in our system. Index vectors of all documents are encrypted before outsource to the cloud. DO create query vector based on keywords entered by Data user. To provide user privacy, query vector encrypted, as Trapdoor and send to Data user. The Data owner sends search access control to the authorized data user.

Data users: Data users are the users who accessing sensitive data from the cloud. The data user sends keywords or synonyms to data owner, which are interested to search. The Data user receives trapdoor and searches access control from data owner and sends trapdoor and access control to the cloud server to retrieve required documents from the cloud.

Cloud Service Provider (CSP): Cloud server receives encrypted documents and encrypted index vectors from data owner and stores into data owner's cloud storage. Cloud server having the capability to take the data request from user and check the search access control of the user. It will retrieve the documents from cloud storage depending upon the privileges to access number of documents. To increase the document retrieval accuracy from cloud server, the top scored (ranked) documents return to data user from the cloud server. Fig. 1 shows the architecture of multi-keyword synonym query over encrypted cloud data.

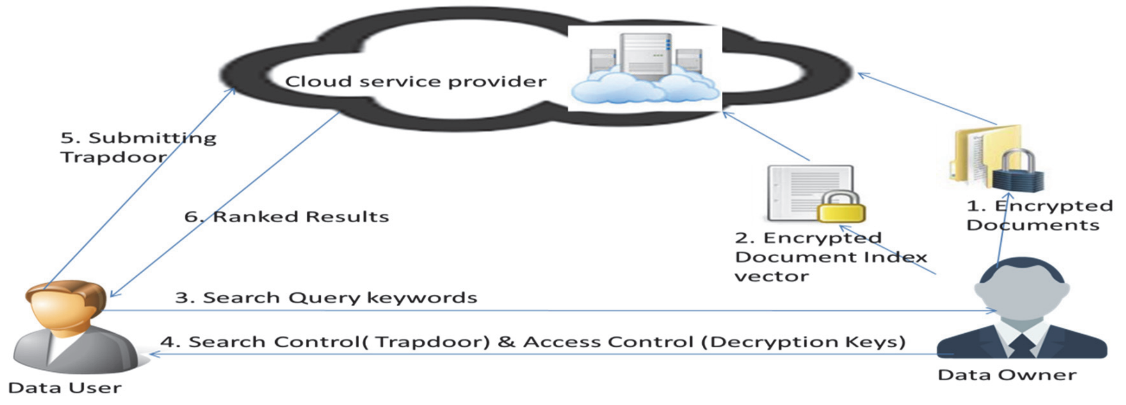


Fig. 1. Architecture of multi-keyword synonym query over encrypted cloud data

Threat model

The cloud server is measured as "honest-but-curious" [15] in our proposed system. The cloud server follows the proposed method specification and also examines data in its cloud storage and data which are received from data user during the processing to learn additional information. We consider one threat model for our system with different attack capabilities that is as follows:

Known ciphertext model: In this model, the cloud server knows only encrypted documents and encrypted index vectors, which are outsourced from data owner.

Design Goals

To enable ranked search for effective utilization of outsourced cloud data that guarantees security and performance as follows:

Accuracy-Improved Multi-keyword Ranked Search: Our system supports multi-keyword search and to retrieve top scored documents instead of all unnecessary documents with more accuracy.

Semantic and Synonym query: It supports retrieval of data based on synonyms entered by data users. It also retrieves semantically related documents using Latent semantic search.

Privacy preserving: It prevents the cloud server from learning information about data user identity and search keywords. The cloud server could not detect the keywords in query, index vector by examining the statistical information like keyword frequency.

IV. PRELIMINARIES

1) Notations:

DC: is a collection of m plaintext document files, as $DC = \{d_1, d_2 \ldots, d_m\}$.

ED: is a collection of encrypted documents stored in cloud storage expressed as $ED = \{c_1, c_2, \ldots, c_m\}$.

D: it is a keyword and synonym dictionary including n keywords and for each keyword t synonyms. It is expressed as $D_{n\times t} = \{k_{11}, k_{12}, \ldots\ldots., k_{1t}$

$K_{21}, \ldots\ldots\ldots.., k_{2t}$

..

..

$K_{n1}, k_{n2}, \ldots\ldots,k_{nt}\}$

Sample Dictionary used in our model is shown in Table I.

W: the subset of keywords and synonyms of D in search query.

F: set of index vectors associated with DC denoted by $F = \{F_1, F_2, \ldots.., F_m\}$.

I: set of encrypted index vectors for F, denoted by $I = \{I_1, I_2, \ldots., I_m\}$.

TF: The sum of keyword frequency and corresponding synonyms (in the dictionary) frequency in the document. In general Term frequency means number of times the term appears in the document.

L^1 [j]: the index vector for document d_j using LSI where the data in $L^1[\,i\,][\,j\,]$ represents the term Frequency $TF_{i,j}$ of the corresponding keyword or synonym W_i in document d_j.

Q: query vector includes the keywords or synonyms interested to search by data users

T: Trapdoor for a set of keywords in W based on query vector Q. It is encrypted form of query vector.

E_k: top ranked documents list according to their relevance to W. It is a descending order list based on the documents scores.

Table I: $D_{7\times5}$ Dictionary with keywords and corresponding synonyms

D[i][j]	**Keyword**	**Synonym1**	**Synonym2**	**Synonym3**	**Synonym4**
	1	2	3	4	5
1	Privacy	Secrecy	Confidentiality	Solitude	Isolation
2	Rank	Rampant	Fecund	Overgrown	Vigorous
3	Document	File	Text	Article	Essay
4	Frequency	Occurrence	Rate	Regularity	Incidence
5	Term	Word	Period	Tenure	Stint
6	Trapdoor	Access	Entrance	Doorway	Hatch
7	Information	Material	Data	Info	Evidence

2) The Vector Space Model for Relevance Score:

The representation of a pool of documents as vectors in a common vector space is called vector space model [13]. This model along with TF-IDF is used in plain text information retrieval, which supports ranked multi-keyword search efficiently. Here, $TF_{t,d}$ is the term frequency defined as the number of times keyword (term) t appears in the document d. The number of documents that contains keyword t is known as document frequency (DF_t). Inverse Document Frequency (IDF_t) of a keyword t obtained from the total number of documents is divided by document frequency. That means, TF denotes the importance of keyword in the document file and IDF specifies the degree of distinction in the whole document file collection. Every document file is denoted with a vector, whose elements are the normalized TF weights of keywords in this document file. The Query is also denoted with a vector, whose elements are normalized IDF weights of query keywords in the document collection. Naturally, the size of the query vector and document vector are equal to the total number of keywords in the dictionary. The dot product between query vector and document vector gives the relevance score of the corresponding document; it quantifies the relevance between the query and the corresponding document.

$TF_{t,d}$ weight $(w_{t,d}) = \log (TF_{t,d})$
IDF_t weight $(w_{t,q}) = \log (N/ DF_t)$
Normalized $TF_{t,d}$ weight $= \frac{w_{t,d}}{\sqrt{\sum_{t=1}^{N}(w_{t,d})^2}}$
Normalized IDF_t weight $= \frac{w_{t,q}}{\sqrt{\sum_{t=1}^{N}(w_{t,q})^2}}$

Relevance score of query vector Q and document vector or index vector F_d is calculated as follows:

Score $(F_d, Q) = \frac{\sum_{t=1}^{n} w_{t,d} \cdot w_{t,q}}{\sqrt{\sum_{t=1}^{N}(w_{t,d})^2} \cdot \sqrt{\sum_{t=1}^{N}(w_{t,q})^2}}$

3) Latent Semantic Index (LSI):

LSI [5] is a technique in natural language processing, in particular in vector semantics, of examining associations between a set of documents and the keywords they contain by generating a set of ideas associated with the documents and keywords. LSA adopts those keywords that are close in meaning and will follow in similar parts of the text. A matrix holding keyword counts per section of text (rows denote unique keywords and columns denote each section of text) is created from a large part of text and a mathematical method called Singular value Decomposition (SVD) is used to moderate the number of rows while conserving the similarity organization among columns. Keywords are then matched by taking the cosine value between the two vectors (or the dot product between the normalizations of the two vectors) made by any two rows. Values close to 1 denote much related keywords while values close to 0 denote very dissimilar keywords.

In this paper, the keyword-document matrix consists of m rows, each of which denotes the index vector for each document, as demonstrated in the equation(1).

$$L = (L [1], L [2], L [3], \ldots\ldots\ldots, L [m]) = \begin{bmatrix} TF11 & \ldots\ldots\ldots\ldots. & TF1n \\ \vdots & \ddots & \vdots \\ TFm1 & \ldots\ldots\ldots\ldots & TFmn \end{bmatrix} \quad (1)$$

Then, we take a large keyword-document matrix and divide it into a set of orthogonal elements from which the original matrix can come close to linear permutation. For example, a keyword-document matrix can be divided into the product of three matrices. Decompose matrix L and find the U, S and V matrices, where

$$L_{m\times n} = U_{m\times t} . S_{t\times t} . V^T_{t\times n}$$

Such that U and V^T have orthonormal columns, S is a diagonal matrix. This is termed as singular value decomposition (SVD) of L. The diagonal values in S are termed as singular values, the first *k* largest singular values in S need be kept and the new size of S is k×k, delete remaining values from S. Hence we consider *k* columns of S which contains largest values, and then deleting the corresponding K columns of U and V respectively. The multiplication of the resulting matrices is a matrix L^1 which is only approximately equal to L, and it is a rank k matrix.

$$L^1_{m\times n} = U_{m\times k} . S_{k\times k} . V^T_{k\times n} \approx L$$

4) Secure Relevance scores:

To compute a score for each document in the cloud, we adapted inner product from k- nearest neighbor method [16] which provides privacy preservation. In this method, it calculates the distance between two database points from query point, so that it returns nearest results from the query. In this method, it finds k nearest results from the query. The technique which is used is splitting against to known plain text attack Here, we represent the index vector as i for document d_i , the query vector as q for data users, and the secret key is a combination of one *n* bit vector as S and two $n \times n$ inventible matrices as $\{M_1 ,M_2\}$. Query vector q is split into two random vector as $\{q', q''\}$ i is split into two random vector as $\{i', i''\}$. To split a vector into two vectors based on the values of bit vector S. We consider a jth bit of S is 0 then, $i'[j]$ and $i''[j]$ are set as the same value as $i[j]$, whereas $q'[j]$ and $q''[j]$ are set in the 0 or 1 so that their sum is equal to the value of $q[j]$. If the value of jth bit is 1 then the process is applied in reverse for i and q. The splitting index vector pair $\{i', i''\}$ is encrypted as

Encrypt (i) = $\{M_1^T \cdot i', M_2^T \cdot i''\}$

And the split query vector pair $\{q', q''\}$ is encrypted as

Encrypt (q) = $\{M_1^{-1} q', M_2^{-1} q''\}$.

The formula to calculate the score is,

Score=encrypt (p) × encrypt (q)

$= \{M_1^T \cdot i', M_2^T \cdot i''\} \times \{M_1^{-1} q', M_2^{-1} q''\}$

$= M_1^T \cdot i' . M_1^{-1} q' + M_2^T \cdot i'' . M_2^{-1} q''$

$= i'^T . M_1 \times M_1^{-1} . q' + i''^T . M_2 \times M_2^{-1} . q''$

$= i'^T q' + i''^T q''$
$= i^T q$

Clearly, the score is not affected by encryption procedure. Without prior knowledge of procedures, neither query vector nor index vector can be predicted by analyzing their corresponding cipher text.

V. SMSRQE DESIGN

In this section, we give a detailed framework of our proposed scheme. We initially propose to employ Latent Semantic Indexing to implement the secure semantic based multi-keyword synonym ranked search. Data owner has collection of m plain text documents DC = $\{d_1, d_2..., d_m\}$ to outsource. Before outsourcing, he has performed encryption on documents using any symmetric key encryption technique. The cloud server can perform searches on encrypted index vector using our scheme. To do this, the data owner first builds a searchable index vector I from set of keywords and synonyms extracted from the collection of documents. The keyword extraction from documents and synonym extension process is as follows:

Construction of Keyword Dictionary extended by Synonyms:

Now a day, data users are searching interested data rather than all the data from the cloud. Keywords need to be extracted from the document files, In order to search interest data before outsourcing. We adopt the TF-TDF (term frequency inverse document frequency) [13] to extract keywords from documents efficiently. $\text{TF-IDF}_{t,d}$ weight of keyword t in document d is calculated as follows:

$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \text{IDF}_t = \log(\text{TF}_{t,d}) \times \log(N/\text{DF}_t)$

TF-IDF weight is high when keyword t occurs many times with in a less number of documents. It is lower, when keyword t occurs fewer times in a document or occurs in many documents. Its value is lowest when the term occurs almost in all documents. So we used this technique to extract keywords from every outsourced document file. The Dictionary is constructed with n highest TFIDF keywords among all the keywords in document files. In order to achieve better synonym based search for outsourced data, the keyword dictionary needs to be extended by common synonyms. The New American Roget's College Thesaurus (NARCT) [14] is used to build a dictionary with keyword and corresponding synonyms in every row of the dictionary. In the dictionary, the first column indicates keyword and remaining columns indicates synonyms. Example dictionary for our system is shown in Table I.

According to LSA definition, Data owner builds keyword – document matrix L and it is decomposed into three matrices. And then, we diminish the dimensions of the original matrix L to get a new matrix L^1 which is calculated the best "diminished-dimension" approximation to the original keyword -document matrix. With t keywords or synonyms of data user interested to search in W, one binary vector q is created and is denoted as query vector. Where each q[j] =1 that indicates w_j is in D otherwise 0. The similarity score is calculated by the inner product of data vector $L^1[j]$ and query vector q. $L^1[j]$ denotes the jth row of matrix L^1.

The proposed system SMSRQE consists of the following phases:

Setup

Our system initializes in this phase. The data owner generates Secret Key. DO generate n-bit random vector S and two n×n invertible matrices M_1, M_2, Hence the secret key SK is a form of three tuples SK = $\{M_1, M_2, S\}$.

BuildIndex (DC, SK, L^1)

Data owner extracts keywords from document collection and creates a dictionary with keywords and synonyms of each keyword respectively, and then creates an index vector for each document. He forms keyword – document matrix L from index vector like above explained. The matrix L decomposed into three matrices U, S, and T such that $L_{m\times n} = U_{m\times t} . S_{t\times t}. V^T_{t\times n.}$ This is according to latent semantic structure. To reduce the dimension, find out Eigen matrix S from that consider k columns. Delete k columns from U and V respectively. The new three matrices are $U_{m\times k}$, $S_{k\times k}$, and $V^T_{k\times n,}$ if we multiply three matrices we will get L^1, it is approximately same as L. To provide privacy for data, it is necessary to encrypt the matrix L^1 before outsourcing. Now split the each row in the matrix L^1 into L_1^1 [j] and L_2^1 [j] using S bit vector as explained above. The encrypted index vector for jth document is $I_j = \{M_1^T. L_1^1 [j], M_2^T . L_2^1 [j]\}$. The similar process applies to all the documents to generate encrypted index vectors.

Trapdoor (W):

The interested keywords entered by a data user of W, this function generates a bit vector q based on keywords or synonyms in D, if keywords available in D then set 1 to a specific dimension in q respectively otherwise set 0. Now apply the same splitting technique which we have applied above to q, divide into q ′ and q ″. to provide search privacy, encrypt query vector before outsourcing. The encrypted query vector is $\{M_1^{-1} q', M_2^{-1} q''\}$.

Query (T, I_j, K):

The query is processed using the inner product of trapdoor T and encrypted index vector I_j by cloud server, and then we will get a score for the jth document in DC. The same inner product is applied for all index vectors, so that it generates scores for all documents. Now it will calculate top K rank scored documents, return to the data user. The similarity score calculation of the document is as follows:

$$
\begin{aligned}
I_j.T &= \{M_1^{T}\cdot L_1^{1}[j], M_2^{T}\cdot L_2^{1}[j]\} \times \{M_1^{-1} q', M_2^{-1} q''\} \\
&= M_1^{T}\cdot L_1^{1}[j].\, M_1^{-1} q' + M_2^{T}\cdot L_2^{1}[j]\, .\, M_2^{-1} q'' \\
&= L_1^{1}[j]^{T}.M_1 \times M_1^{-1}.\, q' + L_2^{1}[j]^{T}.M_2 \times M_2^{-1}.q'' \\
&= L_1^{1}[j]^{T} q' + L_2^{1}[j]^{T} q'' \\
&= L^{1T}q
\end{aligned}
$$

The above calculation gives a score of jth document. The following algorithm finds the top K ranked document files in descending order (E_k).

```
Algorithm 1: Search_TopK_Documents (T, I, K)
Begin
Input: Trapdoor T, Encrypted Index vector I, Score (.), rank K
Output: top k ranked documents $E_k$ in descending order
/* $A_k$ is the kth document in $E_k$ */
For i=1 to m do
   If ($|E_k| < K$) then
        Insert the document file $I_i$ into $E_k$ according to Score (T, $I_i$).
   Else if (Score (T, $I_i$) > $A_k$) then
             Delete the kth document file from $E_k$ and insert document file $I_i$ into $E_k$ according to Score (T, $I_i$)
        End if
   End if
End for
Return $E_k$
End
```

The sample of relevance scores of documents for some input keywords from our experients is shown in Table II.

Table II: The Sample relevance scores of documents for multi-keyword synonym search.

Keywords and synonyms			**Privacy, MRSE, MKQE, security (W_i)**		
Documents	d_1	d_2	d_3	……	d_m
Relevance scores	36.3356868	4.6722820	11.7495305	….	-0.0627363

VI. Experimental Evaluation

In this section, we demonstrate the performance of our method thorough experimental evaluation based on our own created dataset. We implemented MRSE_I and MRSE_I_TF [12] along with our scheme for comparing the performance of our methods. Our scheme developed in two versions first one SMSRQE, which includes keywords and corresponding synonyms in dictionary, second version is SMSRQE_LSI, which includes latent semantic indexing for the first version. The total four schemes are developed in Java jdk1.8.0_40 and executed in jre1.8.0_40 in the system contains windows 7 operating system, Intel® core (TM) Duo processor 1.83GHz and 1 GB RAM. Jama.jar package used to implement Singular Value Decomposition in our scheme. The data types of the elements in a dictionary, index vectors, query vector and invertible matrices are double. The data owner system is installed with MySQL to store dictionary, matrices M1, M2, and bit vector S and documents list, so that whenever a data user requested for trapdoor, it is generated using the same data, which is available in the database. The encrypted index vector for each document and trapdoor (encrypted query vector) are stored in text files, so that cloud service provider generates scores for each document using these text files.

1) Efficiency

Index building:

To build an index vector for each document in the data set, the first step is to extract keywords and synonyms using above discussed TFIDF method and construct the dictionary. Map keyword and synonym set to build index vector for each document-using dictionary followed by encrypting every index vector. The cost of time for mapping and encryption is depends on the size of the data vector which is determining from size dictionary that is the number of keywords and corresponding synonyms. The total time cost for building whole index vector related to the total number of documents in the data set. Fig. 2 (a) shows that, the given same size dictionary n=40×3, cost of time for building whole index vector is linear with the size of the documents in the data set. Time for SMSRQE_LSI is taking a little more time when compared to other schemes because in SVD there is a condition that the size of the dictionary is greater than or equal to the size of the data set. For example, when we take documents m=50 then we should take the dictionary size also n=50×3. Fig. 2 (b) determine the relationship between a dictionary size and cost of time for building index vector with the same size of data set m=40. The major computation for building index includes the splitting process, transpose of matrices M1, M2 and two multiplications of n×n matrix with n size data vector where n is the size of the dictionary. Our proposed methods SMSRQE and SMSRQE_LSI give more efficient and accurate results even though they take little more time when compare to MRSE_I and MRSE_I_TF. Index building is one-time work of data owner before outsourcing to the cloud. It takes time complexity of O (mn^2) for all the four schemes.

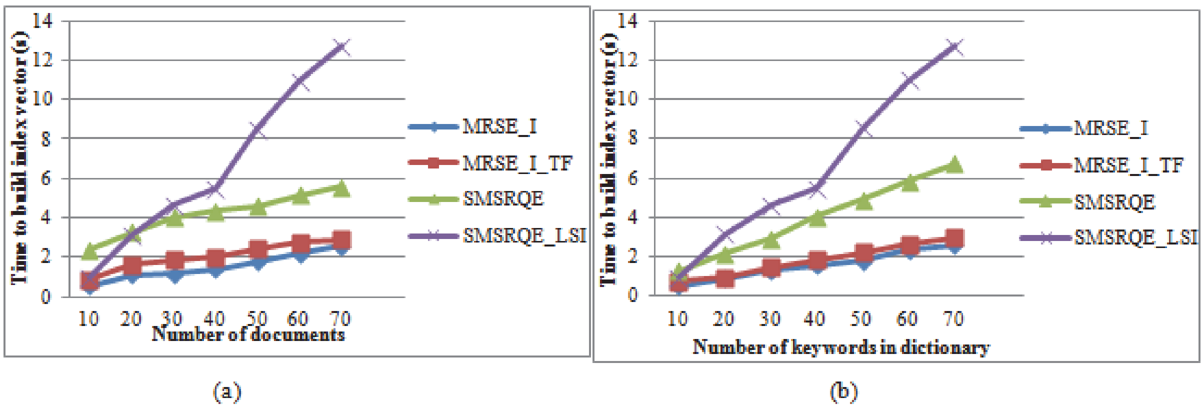


Fig. 2. Cost of time for building index (a) for different sizes of documents with same dictionary size n=40×3 (b) for the same size documents m=40 with different number of keywords in dictionary.

Trapdoor generation:

Cost of time to generate trapdoor greatly affected by the number of keywords in the dictionary and the query, but not number of documents in the data set. Fig. 3 (a) shows time to generate trapdoor with different sizes of dictionary while the same number of documents in the data set m=40. The computation for generating trapdoor includes two matrix inverse, split the query vector into two, and two multiplications n×n matrix and query vector. The time complexity for generating trapdoor is O (n^2) for all four schemes. Results show that trapdoor generation takes less time when compared to index building because index should be built for all documents whereas trapdoor is only one. Fig. 3 (b) shows time to generate trapdoor with different sizes of query keywords while the size of the dictionary is same n=40. Number query keywords have little influence in time when compared to the number keywords in the dictionary.

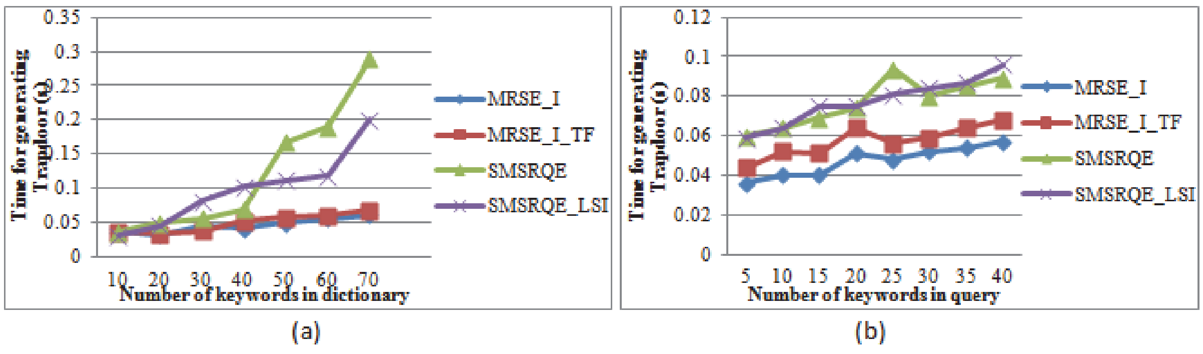


Fig. 3. Cost of time to generate trapdoor (a) for the same query keywords t=10 with different number of keywords in the dictionary (b)for the same dictionary size n=40×3 with different query keywords.

Query Execution:

Cloud service provider executes the query consists of computing similarity scores and ranking of all documents in the data set. The time complexity for similarity calculation of all documents is O (mn) for all four schemes. Fig. 4 (a) shows the time cost of query execution for different number of query keywords with same number of documents m=40. MRSE_I and MRSE_I_TF take almost equal cost because they have the same dimensionality. The Figure showing that impact of query keywords on query execution is very less that is why it is almost horizontal line. SMSRQE and SMSRQE_LSI take little more time because the computation cost is more but, it gives efficient and accurate results. Fig. 4 (b) shows that time cost of query execution for different number of documents with the same number of query keywords t=10. The curve is linear for different number documents in the data set because the number of documents increase then score calculation also takes place for those entire documents in the data set.

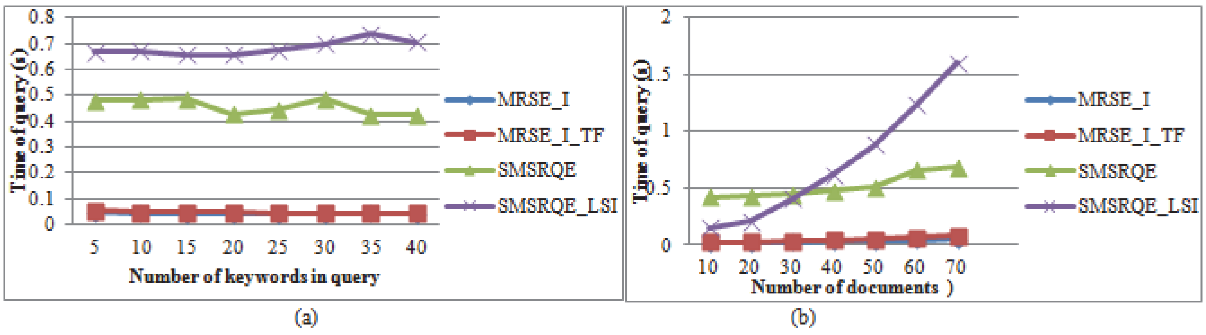


Fig. 4: Cost of time for query execution (a) different number of query keywords with the same number of documents m=40 (b) different number of documents with the same number of query keywords t=10.

Performance of Top K Retrieval:

Top k highest scored documents retrieved in the retrieval phase. Fig. 5 (a) shows the cost of time for top k=10, k=20, and k=30 retrieving documents from the cloud server for different number of documents with the same query keywords t=10. The Figure shows the time cost is almost equal for k=10, k=20, and k=30 because the difference in k value is very less and it is linear, when the number of documents increases then, the cost of time also increases linearly. If we take k=100, k=300, and k=500 then time cost is differ for different number of documents. Fig. 5 (b) shows time cost for retrieving top k=20, k=30, and k=40 for different number of query keywords with the same number of documents m=40. The graph is an almost horizontal line because the impact of query keywords on retrieval cost is very little. The Time cost is almost same for all k=20, k=30, and k=40 because the difference in k value is very less.

2) Precision and Rank privacy

To know the accuracy of the search results N. Cao et al[12] defined precision as $P_k=k'/k$, where k' is the number real top-k documents in k returned documents and k is the number of ranked documents returned by the cloud service provider. We adapted precision to evaluate search results of our proposed scheme. The precision is calculated for all four schemes with the same data set and compared in Fig. 6a. The Fig. 6a shows that our schemes are more accurate than MRSE_I and MRSE_TF. N. Cao et al [12] also defined rank privacy at point k is the average rank perturbation $\widetilde{p}_k$ for each document f in the returned documents expressed as $\widetilde{p}_k = \sum \widetilde{p}_d/k^2$. Where $\widetilde{p}_d = |r_d - \widetilde{r}_d|$, r_d is the rank number of document d in returned top-k documents, and $\widetilde{r}_d$ is the rank number in the real ranked documents. Fig. 6b shows the rank privacy of all four schemes with comparisons. Our proposed scheme exhibits more rank privacy when compare to the MRSE_I and MRSE_TF schemes.

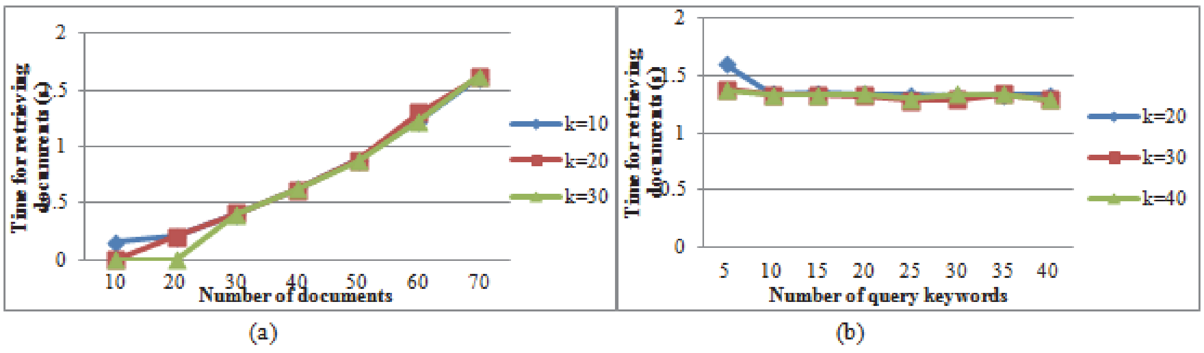
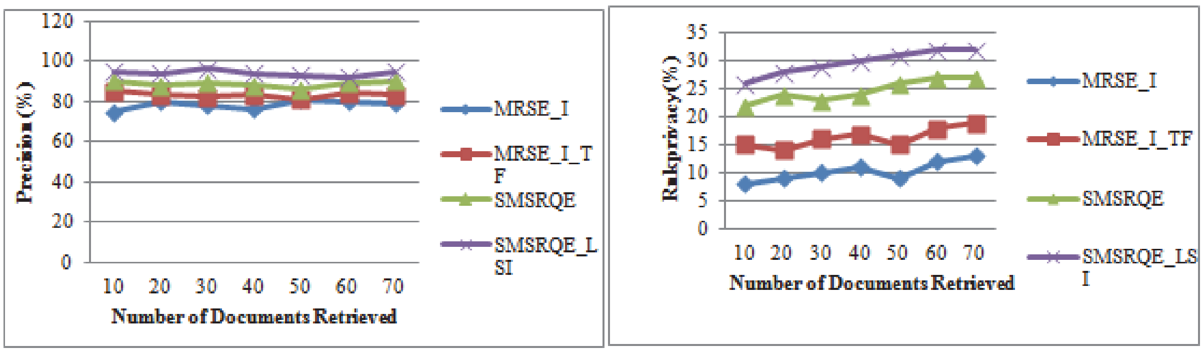


Fig. 5. Cost of time for top K retrieval (a) for different number of documents with same query keywords t=10 (b) for different number of query keywords with the same number of documents m=40.



(a) (b)

Fig. 6. (a) precision (b) rank privacy

VII. Conclusion

In this paper, we intend to provide feasible solutions for semantic multi-keyword synonym ranked query problems over encrypted cloud data in the cloud computing environment. Our main contributions in this paper are four aspects: the first is semantic search based on latent semantic index, the second is a multi-keyword search, the third is synonym-based search and the last one is top k ranked search. Our experimental analysis showing the query results is more efficient and accurate when data user enters synonyms of predefined keywords or keywords due to lack of exact knowledge about the data. Experiment results show that our schemes give more efficient and accurate results when compared to MRSE_I and MRSE_TF. The future work is to research semantic based searching over encrypted cloud data that supports natural language processing techniques. In our scheme, score calculation for each document is linear, it takes costlier, and to avoid this in the future we will use logarithmic algorithms.

References

[1] C. Wang, "Secure ranked keyword search over encrypted cloud data", Distributed Computing Systems, 2010 IEEE 30th International Conference, IEEE, 2010.
[2] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data", INFOCOM, 2011 Proceedings IEEE, 2011.
[3] C. Wang, Kui Ren, Shucheng Yu, Urs, K.M.R "Achieving usable and privacy-assured similarity search over outsourced cloud data", INFOCOM, Proceedings IEEE, 2012.
[4] C.Yang, Weiming Zhang, Jun Xu, Jiajia Xu"A Fast Privacy-Preserving Multi-keyword Search Scheme on Cloud Data", Cloud and Service Computing (CSC), 2012 International Conference, IEEE, 2012.
[5] S.T.Dumais, "Latent semantic indexing (LSI) and TREC-2", NIST SPECIAL PUBLICATION SP, pp. 105-105, 1994.
[6] M.Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data", Distributed Computing Systems Workshops, 2011 31st International Conference, IEEE, 2011.
[7] S. Deshpande, "Fuzzy keyword search over encrypted data in cloud computing", World Journal of Science and Technology, vol. 2, no. 10, 2013.
[8] D.X.Song,D. Wagner and A.Perrig,"Practical techniques for searches on encrypted data, In Security and Privacy", 2000. S&P 2000, IEEE.
[9] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Gener. Comput. Syst. 28 (3), pp. 583–592, 2012.
[10] C. Wang, N. Cao, K. Ren, and W. Lou, “Enabling secure and efficient ranked keyword search over outsourced cloud data,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, pp. 1467–1479, 2012.
[11] B. Wang, S. Yu, W. Lou, and Y. T. Hou, “Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud” in Proceedings of INFOCOM. IEEE, 2014.
[12] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy-preserving multi-keyword ranked search over encrypted cloud data,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 1, pp. 222–233, 2014.
[13] Christopher D. Manning, Prabhakar Raghavan, and H. Schütze, Introduction to information retrieval. Cambridge university press Cambridge, England 2009, vol. 1.
[14] P. D. Morehead, The New American Roget’s College Thesaurus in Dictionary Form, 3rd ed., Signet Press: Seattle, pp. 10-900, 2002.
[15] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing,” Proc. IEEE INFOCOM, 2010.
[16] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis, “Secure kNN Computation on Encrypted Databases,” Proc. 35th ACM SIGMOD Int’l Conf. Management of Data (SIGMOD), pp. 139-152, 2009.
[17] Veerraju Gampala, Srilakshmi Inuganti, Satish Muppidi, “ Data Security in Cloud Computing with Elliptic Curve Cryptography,” International Journal of Soft Computing and Engineering (IJSCE),Volume-2, Issue-3, July 2012.
[18] Veerraju Gampla, Sreelatha Malempati, “An efficient Multi-Keyword Synonym Ranked Query over Encrypted Cloud Data using BMS Tree,” International Journal of Applied Engineering Research, vol. 11, no. 1, pp. 781-786, 2016.