Multimed Tools Appl (2018) 77:10135–10156
DOI 10.1007/s11042-017-5055-5

Split keyword fuzzy and synonym search over encrypted cloud data

Raghavendra S[1] · Girish S[1] · Geeta C. M.[1] · Rajkumar Buyya[2] · Venugopal K. R.[1] · S. S. Iyengar[3] · L. M. Patnaik[4]

Received: 28 February 2017 / Revised: 31 May 2017 / Accepted: 24 July 2017 /
Published online: 7 September 2017

Abstract A substitute solution for various organizations of data owners to store their data in the cloud using storage as a service(SaaS). The outsourced sensitive data is encrypted before uploading into the cloud to achieve data privacy. The encrypted data is search based on keywords and retrieve interested files by data user using a lot of traditional Search scheme. Existing search schemes supports exact keyword match or fuzzy keyword search, but synonym based multi-keyword search are not supported. In the real world scenario, cloud users may not know the exact keyword for searching and they might give synonym of the keyword as the input for search instead of exact or fuzzy keyword due to lack of appropriate knowledge of data. In this paper, we describe an efficient search approach for encrypted data called as Split Keyword Fuzzy and Synonym Search ($SKFS$). Multi-keyword ranked search with accurate keyword and Fuzzy search supports synonym queries are a major contribution of SKFS. The wildcard Technique is used to store the keywords securely within the index tree. Index tree helps to search faster, accurate and low storage cost. Extensive experimental results on real-time data sets shows, the proposed solution is effective and efficient for multi-keyword ranked search and synonym queries Fuzzy based search over encrypted cloud data.

Keywords Cloud computing · Data privacy · Keyword search · Searchable encryption · Synonym search

✉ Raghavendra S
raghush86@gmail.com

[1] University Visvesvaraya College of Engineering, Bengaluru, India

[2] Cloud Computing and Distributed Systems (CLOUDS) Lab, Department of Computing and Information Systems, The University of Melbourne, Melbourne, Australia

[3] Florida International University, Miami, FL, USA

[4] National Institute of Advanced Studies, Bengaluru, India

1 Introduction

Explosive growth of data organization in the age of Information Technology has led to acute storage and maintanence problem. Cloud computing helps customers to store data in the cloud on pay on-demand basis, so that data owners need not worry about data storage space and maintenance. Data owners may not have faith in Cloud Service Providers (CSPs), hence they encrypt their data and then outsource the data into the cloud, so that other users cannot understand the encrypted data. It is not just the problem of data storage, but the important thing is keyword search on encrypted text rather than on plain text. User can download the data into the local machine from the cloud space, decrypt and then search over plain text. In practical, the performance is degrades with utilizes more bandwith. Hence, the challenge is to search the keyword over encrypted data within the cloud storage itself.

As cloud computing is easily accessible, the sensitive data being stored or outsourced into the cloud, like banking sector records, profiles of social networking sites, data of public sector like army, data of government research organizations, medical records and so on needs protection. The fact is, that data stored in the cloud server is not safe even though Cloud Service Providers (CSPs) use firewall mechanism for data security. CSPs may share the data of one user with another user who has also stored the data in the same cloud server or some employee belonging to CSP may access the information stored in cloud server for unauthorized purpose or to leak somebody's data to another user illegally. So, it is always recommended for data owners to encrypt the data and then outsource into the cloud. The encryption of data makes the effective use of data utilization, but the challenge is effective retrieving of relevant files among the large number of files that are stored in the cloud using keyword based search. The user can retrieve only the files of interest using keyword search technique. Considering the large number of data users and a large number of outsourced encrypted data files into the cloud, the keyword search problem is challenging as it has to meet the requirements of performance, system stability and scalability.

As most of the information stored by various organizations is in textual format, all the applications have to support similarity keyword search on the data stored. The user might have used the synonym of some keyword in various places in the file. So, it is important to retrieve the synonym of a keyword being searched along with the exact keyword matching. For example, in a file, the term *college* is stored in some documents and in some other files *institution* is used in the place of *college*. When the user searches for a *college*, it has to also retrieve the files which contain *institution* to the user so that user finds more interesting and relevant results. Though the similarity search and synonym search is possible in plain text, it is also necessary and a challenging problem for searching over outsourced encrypted cloud data due to privacy and security reasons. Ranked search [16, 17, 19] is useful, as it sends back only the relevant data and avoids network traffic. This is quite important as most of the users use pay-as-you-use cloud parameter. Moreover for privacy preserving, these ranking operations should not leak any information related to keyword. The ranking system should support multiple keyword search as well as synonym keyword search to improve the accuracy of the search result.

Motivation Once the data is stored in cloud, the most important operation is to search. Even though the data is accessible only to a few authorized users, who perform search operations frequently, search operation should preserve the privacy of users as well as of data and it should also return efficient results. To achieve the privacy of data during search, many methods or techniques for privacy-preserving have been studied. Most of the work focus on

single keyword and multi-keyword search [3, 22, 25], while few works focus on fuzzy keyword search [23, 24, 29, 32]. Few techniques have also been proposed for similarity search on text [26] and on images [31]. The ranked search scheme helps the users to find out the most relevant documents.

Contribution In this paper, to meet the challenge of effective search system. A flexible searching technique has been proposed in this paper which is efficient that supports fuzzy keyword, multi-keyword and synonym based keyword search. This protocol generates index using inverted indexing where the keyword is mapped to documents. This inverted index scheme provides the technique for scoring the search results. If the number of a keyword being searched map to a large number of individual documents, then that is considered as a relevant document. The use of balanced binary tree improves the search efficiency. The relevant documents can be retrieved by traversing the tree. This search scoring technique uses Term Frequency-Inverse Document Frequency for weighting the results.

The contribution of this paper is summarized as follows:

1. Split Keyword Fuzzy and Synonym Search ($SKFS$) has been proposed, which supports fuzzy based multi-keyword search over encrypted cloud data. When the user enters the keyword to search from the cloud data, the user obtains the fuzzy keyword matching in addition to the files which contain the synonym of few predefined keywords.
2. SKFS fulfills the functionality of secured keyword search which maintains keyword privacy; relevant files based on fuzzy keyword and synonym keyword search are retrieved.
3. SKFS ensures that search results using synonym based keyword searches over encrypted cloud data is authenticated by using index-based tree structure.
4. Extensive experimental result shows the effectiveness and efficiency of the SKFS scheme.

Organisation: The rest of the paper is organized as follows: Section 2 presents the reviews of the features of related work. Section 3 briefly describes the necessary background for the techniques used in this paper. The design goals are explained in Section 4. Section 5 describe the overview of proposed multi-keyword ranked search scheme which supports synonym query and our proposed scheme. Performance analysis for index time construction, storage cost of index and search time and security analysis is presented in Section 6. Section 7 contain the conclusions.

2 Related works

Fuzzy keyword search are investigated in [5, 6, 10, 18, 20, 27, 28, 35]. Xu et al. [32] takes linear time to store the searchable cipher-text as keywords and resists keyword guess attack but unaffordable for large database. Tuo et al. [23] propose a semantically secure fuzzy keyword search scheme using the bloom filter which translates the keyword into attribute set and uses independent hash functions to map the elements to some random number using bilinear mapping technique. This method extends from fuzzy identity encryption scheme to fuzzy keyword search scheme but does not support multi-keyword search and synonym search. Chuah et al. [5] introduce a b^{ed} tree index tree construction and storage cost is efficient compared to the symbol-based trie-traversed based and listing based approach. Wildcard-based fuzzy set construction technique is used in [10, 13, 27, 35]. Jie et al. [10]

use dictionary based schemes to remove unwanted keywords and results in relatively small constructed index.

Wang et al. [28] achieve similarity search for top-k ranked keywords and it uses Keyword Fingerprint Extraction which converts a string into a fingerprint vector. The k-Nearest Neighbors(kNN) encryption provides two tiers of protection for keywords being searched but does not provide support for synonym search. Fu et al. [6] use vector space model to construct the index for document; a balanced binary tree-based index structure is used for searching the keyword. Synonyms of predefined keywords are searched but this requires increased storage space. Shekokar et al. [20] designed a privacy-preserving fuzzy keyword search to achieve effective usage of encrypted data stored remotely in cloud. Indexing technique is not used for mapping the keywords and takes more search time. In [35] weighted ranking algorithm has been used to compute the weight of each word in the fuzzy set, that involves high computation overhead.

Different privacy-preserving search method over encrypted data [2, 21, 33, 34] had been proposed. Yuan et al. [34] use two algorithms, Index Generation to generate the index and Document Retrieval to retrieve only top-k documents. The second algorithm i.e., Two-Server Secure Search and Document Retrieval searches keywords from the file server and provides adaptive semantic security. The main advantage of this method is the capability of multi keyword search in a single query. Sun et al. [21] propose search index based on term frequency–inverse document frequency(TF-IDF) and vector space model along with cosine similarity measure which supports multi-keyword search and ranking method. Bijral et al. [2] design a B tree search algorithm that uses inverted index and fully inverted index. This method shows that wildcard based search has more keywords than fuzzy keywords in dictionary based search. It is observed that Dictionary-Based Search method is efficient with respect to time. Xu et al. [33] propose a dictionary-based fuzzy set is constructed for fuzzy keywords and uses coordinate matching which returns all the matches as possible so that the data files obtained is relevant. One-to-many order preserving mapping scheme is used to build the inverted index and for searching.

Wang et al. [29] propose a verifiable fuzzy keyword search scheme based on the symbol-tree which supports the fuzzy keyword search and also performs the verifiability of the searching result. This scheme uses two algorithms i.e., Generate Fuzzy Set which generates the Fuzzy keyword set and Searching Tree which generates set of file ids that helps in security and privacy-preserving. Wang et al. [24] propose a scheme for multi keyword fuzzy search which searches by exploiting the locality-sensitive hashing technique. This method does not expand the index file, instead it searches fuzzy keyword matches using algorithmic design where the predefined dictionary is not required and using Symmetric cryptography. The index generation procedure is a single time computation and as the new keywords are inserted, the index generation time increases linearly. The search also happens in the encrypted index. Here, locality-sensitive hashing (LSH) method is used to construct the index of a file and this is efficient to search multiple keywords.

Few methods [4, 7, 11, 12, 14] support semantic search schemes. Khan et al. [11] formulate secure rank search of fuzzy multi-keyword which returns the matching files when input keyword exactly matches the predefined keyword. If it fails to match the exact files, it returns the possible keywords from the dictionary based on similarity semantics. The ranked fuzzy multi-keyword search(RFMS) method have two algorithms Build-Index and Fuzzy-search. As the encryption files contains special characters, the dictionary attack is not possible. Each element of the trapdoor is unique, it is one-to-many mapping between plaintext keyword and its cipher text which contains special characters that results in

enhanced security. Fu et al. [7] propose a semantic search scheme which returns keyword based proper match and also a keyword based semantic match and the search result are verifiable. Whenever the keyword is submitted for search, it builds the term similarity tree and then shortest path and similarity between keywords are calculated. The user can cross check the correctness and completeness of the result obtained. It uses the hash function query to achieve index privacy. Fu et al. [8] propose a semantic keyword based search scheme and privacy preserving. The stemming algorithm is used to construct the stem set and reduce the dimensions of index. A symbol based trie is adopted for construction of index which improves the search efficiency.

Ko et al. [12] propose a semantical scheme which searches data on mobile devices where a user query is translated into a query graph and then retrieved.This method uses the algorithm for finding answer graphs from a query graph, then the answer graph is translated to SQL statements by traversing answer graphs and obtain the result from the database. It overcomes the limitations of keyword based full text search. Chinnasamy et al. [4] propose semantic secure keyword based search scheme(ESSKS) that retrieves exact details needed by the user and ensures that same keyword does not always produce similar results. This method addresses the problem of data integrity while transferring data from the user to cloud and vice-versa. The user has to encrypt a file using secure symmetric encryption that reduces the search time of the keyword and does not allow unauthorized users to access the data. Moh et al. [14] propose a semantic search with three different schemes like synonym-based, wikipedia based and wikipedia based synonym keyword search. This scheme uses Data encryption standard algorithm for symmetric key encryption as well as decryption. The search results are in the form of similarity score on the data file. This scheme performs better than wildcard-based fuzzy set construction scheme with respect to storage requirements, performance and the search result in terms of preserving data security and maintaining privacy.

Balamuralikrishna et al. [1] propose a fuzzy keyword search on encrypted cloud data along with maintaining privacy of the searched keyword. Wildcard and gram based techniques use string matching algorithm. Trie traverse tree structure has been constructed that are transformed from the resulted fuzzy keyword sets. Zhou et al. [35] propose a scheme to generate fuzzy keyword search over encrypted cloud data which uses k-grams index to return fuzzy results where keywords are searched as wildcard queries on plain-text files. Weighted ranking algorithm has been used to compute the weight of each word in the fuzzy set. Xia et al. [30] propose a public key encryption scheme in mobile cloud storage for a group of users for data sharing. A asymmetric group key agreement protocol and proxy re-signature is used for updating the searchable keywords which are encrypted. It ensures that mobile users in some group have to share the common secret key and update it when group members change.

3 Background

Researchers have been working on finding out the efficient way of searching keyword over encrypted cloud data to perform the searchable encryption in cloud computing. Many search schemes have been proposed like fuzzy keyword search scheme which uses wildcard-based technique to generate fuzzy keyword sets. Other schemes use TF-IDF method to find out keyword weight, that can generate top-k relevant data files. But, this supports single

keyword search. A multi-keyword search scheme has been proposed in [21] based on vector space model which supports more accurate results as it stores the weight of each keyword. The authors [9] use searchable index tree which is a balanced binary tree where each internal node is stored with keywords. This scheme takes more space for index storage and it has to traverse a large number of nodes for searching the keyword. Moreover this scheme does not support fuzzy search and semantic based search.

3.1 Security requirements

In the SKFS Scheme, we consider the cloud service provider to be honest-but-curious that means the cloud service provider performs the task allocated by the data owner and search the suitable document based on the user need. In spite of this, it is curious to know the information about the storage files and the generated trapdoors to gain extra information. Explicitly, the SKFS targets to offer the following safety requirements:

- *Content Secrecy and Index Privacy:* Content files and index file must be encrypted before outsourcing to the cloud server. The index privacy will have two safety requirements, i.e., the cloud sever not permitted to learn the information of the index file because the information of the index file straightforwardly reveals the information of the content files. In additional, cloud server cannot assume any relationship among the content files and keywords during exploring the encrypted index file.
- *Query Confidentiality:* The cloud server not permits to learn any information from the query keywords. Two types of privacy requirements are trapdoor privacy and access pattern. Each query keyword produces a trapdoor to access the cloud storage space and search across the encrypted index. The query information is present in the trapdoor but in an encrypted manner. After observing the trapdoor cloud server not permits to learn anything from user's query. Trapdoor privacy can safeguard by query keywords, number of keywords present in the query and the cloud server cannot assume any relationship among the two trapdoors. The cloud server not permits to know the accurate keyword holds in the trapdoor of the user search query. Access pattern refers to the retrieved files based on series of search results. In the SKFS, the access pattern completely hidden from the cloud service provider. The query search results should be vague form every user because the cloud server not allow to know the details of the retrieved files.

4 Problem statement and system model

4.1 System model

We consider the cloud computing architecture with three modules, i.e., Data Owner(DO), Data User(DU) and Cloud Server (CS) as shown in Fig. 1. Data owner has a set of n data files $D = F_1, F_2, F_3, \ldots, F_n$ which are stored in the cloud server. All these files are encrypted and a set of encrypted files $ED = (EF_1, EF_2, EF_3, \ldots, EF_n)$ are generated. The data owner extracts the keywords from the data files D. The data owner then removes the stop words from the extracted keywords to form a set $KW = k_1, k_2, \ldots, k_m$. Next, the data owner obtains the synonyms for keywords $S_1, S_2, \ldots, S_x$ and generates a searchable encrypted index EI for keywords and its synonyms, the encrypted index EI and a set of encrypted files ED are outsourced into the cloud server.

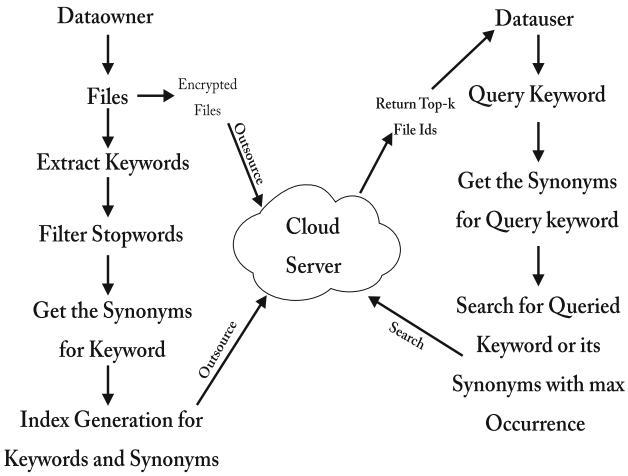

Fig. 1 Index generation on split keyword system architecture

When the authorized user performs the search operation, he or she submits an input keyword for search which is considered as Query KeyWord (QKW). The Data User gets the synonyms $QS_1, QS_2, \ldots, QS_k$ for QKW, then searches for QKW and its synonyms (see Table 1). The Cloud Server(CS) returns the search results as per the following rules.

1. If the Query keyword matches exactly with any keyword stored in the index, the server returns the files which contain the keyword.
2. If any file contains any synonym of the QKW, the cloud server returns also those files.

4.2 Design goals

The design of our system model supports multi-keyword search over outsourced encrypted data in cloud with the following security and performance paradigms.

1. *Synonym Search* : Our search scheme supports fuzzy keyword search as well as synonym search for the input keyword and returns the files containing keyword or its synonyms
2. $Privacy - Preserving$: Our method is designed to meet the privacy challenge and prevents a cloud server and other cloud users from reading any information from any data files or from index being stored
3. $Efficiency$: The above functionalities is achieved with low storage, low network traffic and with low computation and search time.

4.3 Inverted index

Inverted index is an indexing data structure that stores a mapping from keywords to the respective set of files containing the keyword, which allows full text search. By using the inverted index, it is easy to get the files in which a given keyword has maximum number of occurrence. This is usually done by calculating a numerical score based on any ranking function. An example for inverted index is shown in Table 2.

Table 1 Notations

Symbols	Definition
D	The plain-text document collection to be outsourced as a set of n data files $D = (F_1, F_2, F_3, \ldots, f_n)$.
ED	A set of encrypted documents collection to be outsourced as a set of n data files $ED = (EF_1, EF_2, EF_3, \ldots, EF_n)$.
EI	Encrypted Index.
MF	Merge Factor.
F_i	The file identifiers F_i to locate uniquely the actual file.
KW	The extracted distinct keywords from the document collection D, denoted as a set of m keywords $KW = (k_1, k_2, \ldots, k_m)$.
QS_i	Synonyms for each extracted distinct keyword, denoted as $(QS_1, QS_2, \ldots, QS_k)$.
$k_{sf}*$	Keyword denoted as wildcard format.
k_{sf}	Keyword with first sf number of characters.
sf	Split factor to divide the keywords.
$n - sf$	Number of Remaining characters of keyword after splitting it.
k_{n-sf}	Remaining characters of keyword after splitting it.
QKW_{sf}	Query keyword with first sf number of characters.
QKW_{n-sf}	Remaining characters of query keyword after splitting it.
R_i	Set of k_{sf} for many keywords where k_{sf} is same.
k_i	Individual keyword.
QKW	User interest queried keyword.

4.4 Tree based structure

Searchable index is a balanced binary tree. The index tree EI is built from the set of data files $D = (F_1, F_2, F_3, \ldots, F_n)$. The tree is built using the following procedure, which is expressed as *generateIndex(D)* as follows:

1. A leaf node in a tree is generated for each document/data file F_i in D, which stores the file identifier F_i and the index list.
2. Futher, the tree is constructed using postorder traversal with all the leaf nodes generated in first step. Each internal node of index tree contains atleast one element in linked list, where each list stores the details of keywords in wildcard based format where k_i belongs to F_i.
3. The linked list is generated in each internal node. If k_{sf} belongs to F_i, where $k_{sf} = (k_{sf}*)$, it adds the length of * and file F_i into the list; characters of * in k_i is also encrypted and added into the list. Similarly, a list node for each keyword is created.

Table 2 Inverted index matrix

Keyword	File $Id(F_i)$
Computer	F_1, F_5
Network	F_2, F_3
Software	F_1, F_4, F_5
Router	F_3, F_4, F_5

4.5 Tree based search

The sequential search method for keywords in input search is as follows: Procedure starts from the root node and then it searches for an internal node, it checks with QKW_{sf} and k_{sf} in the linked list. If both match, then it searches to match for remaining characters of k_i inside the list. If it is found, searching stops in the subtree, otherwie, it continues to search in the child nodes. When it traverses the node, it gets the file Id and the number of occurrences of the keyword in each file. In this search method, it traverses lower number of nodes as it stores k_i in the wildcard based format and each file $id\ F_i$ of that k_i occurrence in the same list and also because it uses inverted index structure.

5 Proposed work

This section gives the overview of the SKFS scheme. An efficient multi-keyword ranked search scheme using synonyms is designed as follows:

Algorithm 1 SKFS: Split Keyword Fuzzy and Synonym Search

Input : set of data files $D = F_1, F_2, \ldots, F_n$
Output : A encrypted index EI

Index Generation;
Create index for each document separately using generateIndex(D) method;
Merge the group of indices based on Merge Factor MF;
Create a stack for indices;
Create index for all the documents as explained in Section 5.1;
Push new index being created into stack;
Assume merge factor $MF = 5$;
$n = \infty$;
while ($max\_size \geq 1 \&\& max\_size \leq n$) **do**
    read current;
    if *there are MF number of index with max_size docs on top of stack* **then**
        Pop MF number of indexes from stack;
        Merge them into one index;
        Push the merged one into stack;
    else
        Break;
$max\_size* = MF$;

Search Phase
Search($QKW_{sf}, n - sf, QKW_{n-sf}$)
for *(i=1 to $QKW_{sf}[n]$)* **do**
    if *(QKW_{sf}==$k_{sf}[i]$)* **then**
        if *(there exists node with $l_i == n - sf$)* **then**
            if *(QKW_{n-sf} is present inside)* **then**
                Return File set F_i where $QKW \in F_i$
            else
                Break;

1. $Initialization$: This phase is executed by DO to initialize the scheme. It generates secret key sk for encrypting the data before outsourcing into cloud.
2. *Index generation*: The Index phase is executed by DO to build the index using sk and a set of distinct keywords k_i of data files D and outputs the index..
3. *Query generation*: The query function is generated by DU to generate the query out of the keyword being given for search as input. It splits the keyword QKW and its synonym QS_i based on the split factor sf and generates query.
4. $Search$: The search phase is executed by CS to search for the files F_i which contains the keyword QKW or its synonyms QS_i. It takes query and index as input and returns the set of files where the keyword and its synonyms are present

5.1 Steps in the algorithm SKFS

1. The DO generates the secret key for encrypting the data files before they are outsourced into the cloud.
2. In this procedure, A tree based index structure is built for fuzzy keyword set which enables multi keyword search. The DO reads all the data files D and builds a set of distinct keywords for each file. In addition, a set of file identifiers for all files i built and then outsourced to CS. The text documents can be expressed as follows.

$$\begin{aligned} F_1 &= k_1^1, k_2^1, \ldots, k_{n-1}^1, k_n^1 \\ F_2 &= k_1^2, k_2^2, \ldots, k_{n-1}^2, k_n^2 \\ &\vdots \\ F_m &= k_1^m, k_2^m, \ldots, k_{n-1}^m, k_n^m. \end{aligned}$$

5.1.1 Extracting synonyms

$$\begin{aligned} F_1 &= k_1^1, s_1^1, \ldots k_2^1, s_2^1, \ldots, k_{n-1}^1, s_{n-1}^1, k_n^1, s_n^1 \\ F_2 &= k_1^2, s_1^2, \ldots k_2^2, s_2^2, \ldots, k_{n-1}^2, s_{n-1}^2, k_n^2, s_n^2 \\ &\vdots \\ F_m &= k_1^m, s_1^m, ..k_2^m, s_2^m, ..., k_{n-1}^m, s_{n-1}^m, k_n^m, s_n^m. \end{aligned}$$

The fuzzy keyword set generated here is based on the wildcard. Index is created for each document by splitting the keyword based on the split factor sf, which divides the keyword k_i into two tokens k_{sf} and k_{n-sf}, where n is the length of k_i. The indexed linked lists stored in each node contains the list for keyword starting with k_{sf}. The linked list is denoted as $k_{sf}, n - sf, F_i, k_{n-sf}$.

5.1.2 Generating index for individual files

$$\begin{aligned} IF_1 :&= R_i^1 \left(\left(l_{n-i}^1, ER_{n-1}^1 \right) \ldots \left(l_{n-i}^N, ER_{n-1}^N \right) \right); \\ &= R_i^2 \left(\left(l_{n-i}^1, ER_{n-1}^1 \right) \ldots \left(l_{n-i}^N, ER_{n-1}^N \right) \right); \\ &\vdots \\ &= R_i^n \left(\left(l_{n-i}^1, ER_{n-1}^1 \right) \ldots \left(l_{n-i}^N, ER_{n-1}^N \right) \right); \end{aligned}$$

where $R_i + ER_{n-i} = K_i$.

After creating the index for each document, merging of index is done using the merge factor MF. Initially, all the individuals index are stored into stack. Then MF number of indexes are extracted and merged as below.

5.1.3 Merging index files

$$\begin{aligned} I &= \{R_i^1\{\left(l_{n-i}^1, F_n, ER_{n-1}^1\right) \ldots \left(l_{n-i}^N, F_n, ER_{n-1}^N\right)\}; \\ &= R_i^2\{\left(l_{n-i}^1, F_n, ER_{n-1}^1\right) \ldots \left(l_{n-i}^N, F_n, ER_{n-1}^N\right)\}; \\ &\vdots \\ &= R_i^m\{\left(l_{n-i}^1, F_n, ER_{n-1}^1\right) \ldots \left(l_{n-i}^N, F_n, ER_{n-1}^N\right)\}\}; \end{aligned}$$

where

$$F_n = \{F_1, F_2, ..., F_n\}, i \leq j \leq n, K_i \in F_i.$$

$$T = R_i^j\{(l_{n-i}^j, F_n, ER_{n-1}^j)\}$$

The merged index is pushed onto the stack and the merging of indexes is continued till MF number of indexes are available. The index is merged based on the R_i, the files contains R_i make into one group. Each R_i presence in X files but it's not same for all R_i.

3. *Query generation*: When the keyword QKW is given for search, the QKW is split into two tokens based on splitting factor sf used in index generation like QKW_{sf} and QKW_{n-sf}. Then the split tokens are encrypted and query set containing (QKW_{sf}, $n-sf$, QKW_{n-sf}) and its synonyms are formed and sent to the cloud server.
4. If the linked list in any node in the index tree matches with QKW_{sf}, then it searches inside the linked list for $n-sf$. If a internal node in the linked list matches with $n-sf$, it proceeds to search inside that node only for finding out the match for QKW_{n-sf}. If there exists QKW_{n-sf} inside the list, it returns the list of file Ids where the keyword contains QKW and its synonyms.

5.2 Algorithm example

In the example, we have assumed two files F_1 and F_2 with various keywords. Initially, the index is created for first file. Each keyword is divided into two sub keywords based on the split factor. The keyword $university$ is divided into $univ$ and $ersity$. While generating the index, all keywords starting with $univ$ are combined together and index is formed as shown in Table 3 i.e., *university, universe and universal*. Let n be the length of keyword, sf the split factor, the index for keyword k_i, first i added to the initial characters of keyword, then the length of remaining characters i.e., $n-sf$ is added. The file id and the remaining characters are added after encrypting. If there are a number of keywords starting with the same characters, then index for each k_i is separated by the delimiter and are added based on length of $n-sf$ in the ascending order.

After the index is created for the index file, the index for the second file is generated in the same manner and both are merged. Check if the index for the keyword is already existing in the index file. If it exists, then only the file id has to be added. In the above example, file $id2$ is added for $university$ and the index for the new keyword $computer$ is also added. In a similar manner, we have to continue adding the new keywords for new files.

6 Performance

Our proposed scheme is verified by implementing the search scheme on the cloud server. We have used real time data sets: National Science Foundation Research Awards Abstracts

Table 3 Example for algorithm

			Number of Files	
			F_1	F_2
Extracted Keywords			University	University
			Universe	College
			Universal	Optical
			Optimize	Computer
			Optical	
Index Format				
File id	R_i	n-sf	ER	frequency
F_1	Univ	6	ersity	4
		4	erse	4
		5	ersal	4
	Opti	4	mize	4
		3	cal	4
F_2	Univ	6	ersity	4
	Opti	3	cal	4
	Coll	3	ege	4
	Comp	4	uter	4
Merged Index File				
R_i	n-sf	ER	frequency	File ids
Univ	6	ersity	4	F_1, F_2
	4	erse	4	F_1
	5	ersal	4	F_1
Opti	4	mize	4	F_1
	3	cal	4	F_1, F_2
Coll	3	ege	4	F_2
Comp	3	uter	4	F_2

1990-2003[15]. Our experiment includes a user and a server. The search scheme is implemented using Java language in Windows machine with Dual CPU running at 1.46 GHz and the encrypted collection of files are stored on the commercial public cloud, Amazon cloud services like *S*3 (simple storage service). The performance is evaluated for index generated time, index storage space, search time and security analysis. Total number of keywords present in index is 94,832 for 10,000 files and 8936 are distinct keywords.

6.1 Index generations time

Figure 2 indicates the time complexity for generating the index which is directly proportional to the number of data files. Although, index generation is performed only once before outsourcing the data into cloud, this operation can be ignored. While generating the index, encryption of the keyword is an additional operation. As shown in Fig. 2, the B-tree method

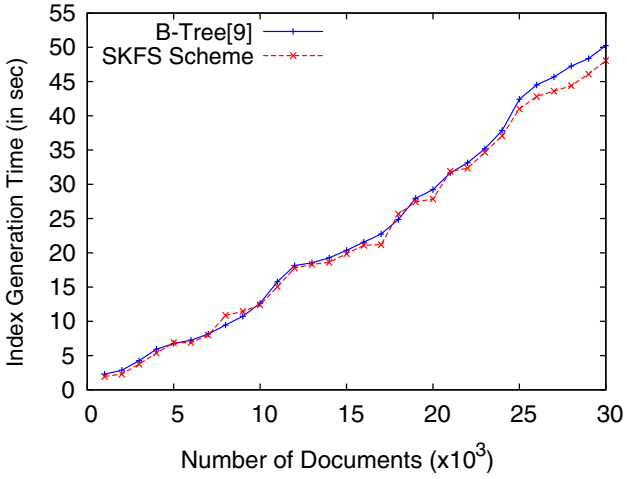

Fig. 2 Index generation time over number of documents

[9] takes 12.64 secs to generate index for 10000 files whereas the proposed scheme takes 12.40 secs to perform the same operation on the same set of data files. The comparison shows that SKFS is already efficient at this stage. Similarly for 20000 files, B-tree requires 29.21 secs and the proposed SKFS scheme requires 27.84 secs. Both the methods grow linearly as the number of files also increments. It can be seen that the index time generation for the proposed scheme is more efficient than the B-tree scheme [9].

6.2 Index storage space

The index file uses less amount of storage space compare to existing scheme. Data security is important rather than for storage space of index. Figure 3 indicates the size of index

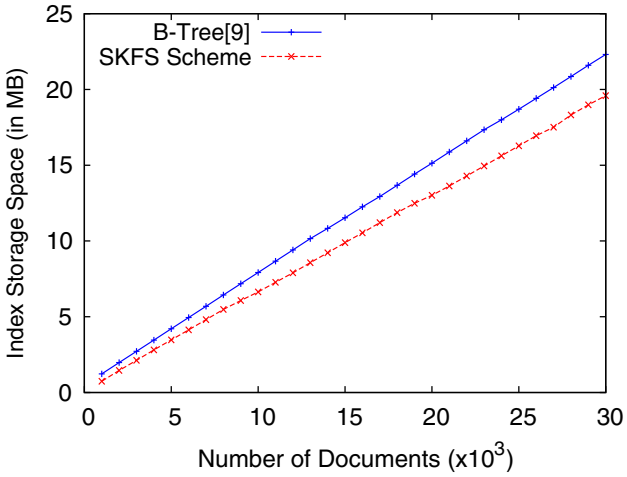

Fig. 3 Index storage space on cloud

tree in both the schemes. The storage cost for 10000 files in B-Tree method is 7.5 MB whereas the proposed scheme costs just 6.33 MB. B-Tree scheme for 20000 files costs 14.43 MB, while the proposed SKFS scheme requries 12.41 MB for the index file. Extensive experiment shows that SKFS scheme takes less space for storage of index file compared to B-tree method [9].

6.3 Search time

In this section, the performance of SKFS search scheme is evaluated with the increase in the number of documents. The search process is initiated by the cloud server. It is done by computing the score of documents and search result is based on the ranking. Figure 4 shows the search time for both the schemes. It shows search time for proposed scheme is efficient compared to the B-tree scheme, as the proposed scheme uses inverted index method and stores the keywords by splitting the index tree. In our scheme, if the first part of query keyword matches with the sub keyword in index, then it compares with the remaining characters in index. Hence, it does not compare all the characters of keyword and so this scheme takes less time for searching compared to the B-Tree. If the first sub keyword in the index does not match in the index tree, it traverses the next node and searches in the next node and the process continues till the leaf node. Let the height of the index tree be $m + 1$ with m leaf nodes and number of nodes is r, the time complexity in B-tree is $O(rlogm)$. In SKFS scheme, the search procedure traverses through one node in each level and hence, the search complexity is $O(logm + 1)$.

6.4 Results acurracy

In the SKFS scheme, synonym and fuzzy keywords are not accurately associated with the queried keyword. Meanwhile some of the false positive keywords are searching to confuse

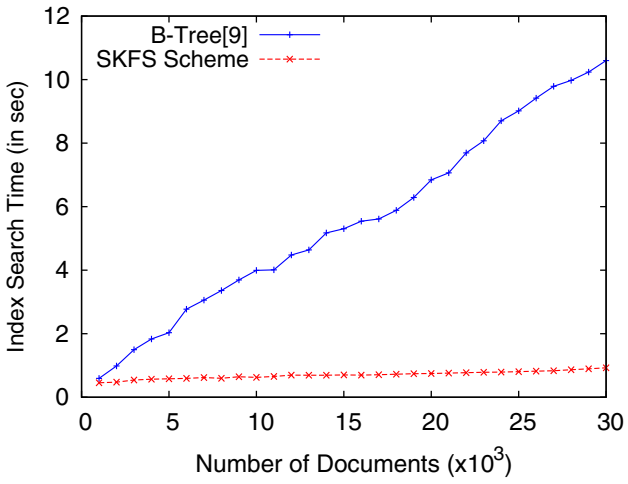

Fig. 4 Search keyword over encrypted cloud data

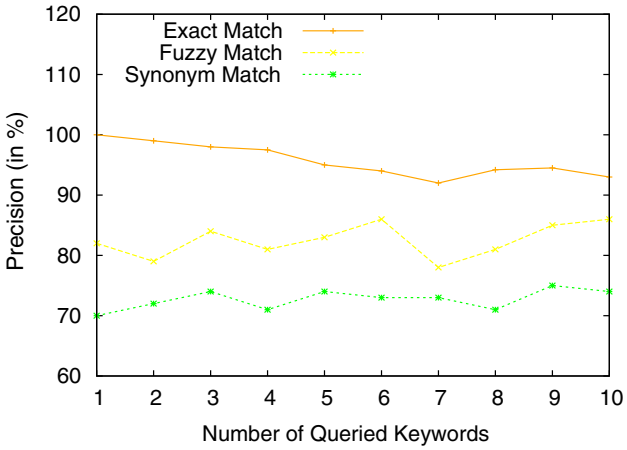

Fig. 5 Precision for synonym and fuzzy keyword queries

the adversary. The precision and recall definition is used to measure the search result accuracy. Explicitly, the precision is defined as

$$Precision = \frac{k'}{k} \tag{1}$$

$$Recall = \frac{k'}{k''} \tag{2}$$

Where k is number of files return from the cloud server and k' is the number of positive top-k response from k returned files. k'' is the number of relevant response from k returned files. The standard deviation $\sigma = 0.5$ have an effect on precision and recall.

In Figs. 5 and 6 Shows the performance metrics of precision and recall variation among exact match, fuzzy match and synonym match. The exact match searches accurate matching

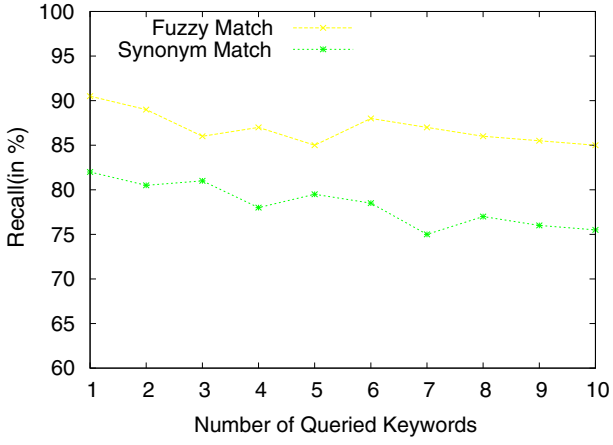

Fig. 6 Recall for synonym and fuzzy keyword queries

keywords and there is no recall for the exact matching because it's not performs on negative search results. The exact match precision is reducing a little from 100% to 92% based on the query keywords rises from 1 to 10. The precision is reduces when increasing the keywords because acceptable amount of negative search results appear in retrieved files. The fuzzy match precision is varies from 78% to 86% based on the query keywords increases from 1 to 10. The fuzzy keyword performs on likely relevance to the exact keyword but the word and spelling might not the accurate keyword. The fuzzy match recall is varies from 90.5% to 85% when the amount of query keywords increases from 1 to 10. The synonym keyword precision is varies from 70% to 75% when the number of query keywords rises from 1 to 10. The synonym keyword performs on a word which has the similar meaning with an alternative word. The synonym keyword set is taken from New American Roget's College Thesaurus in Dictionary Form. The synonym match recall is varies from 82% to 75% when the number query keyword increases from 1 to 10. If the number of keyword increases, precision will increase simultaneously recall will decrease.

7 Security analysis

We examine the security and privacy of the proposed scheme by analysing its fulfilment of security guarantees. SKFS scheme aims to provide Data Privacy to the outsourced files. SKFS Scheme contains: Content Secrecy, Index Privacy and Query Confidentiality. The content secrecy succeeds by using AES algorithm for file encryption before outsourced to the cloud. Index privacy and query confidentiality are the two major security concerns are focused in these work. In index privacy, the index file contains file id, split keywords and frequency of each keyword. Here, we are providing privacy for keywords because the index file access to the cloud server and they are trying to know the content of the file. In query confidentiality, the cloud service providers are trying to know generated trapdoor or access pattern of keyword search request given from the data users.

7.1 Index privacy and query confidentiality

In our SKFS scheme, file content, trapdoor and query keywords are in encrypted form. The secret keys are well secured, the cloud service provider cannot able to assume document details, index keywords and query keywords. The data files stored in the cloud servers are not able to learn or alter any information by unauthorized data user. Index file contains keywords split into two part using split factor; both split parts are encrypted separately and upload to the cloud server. The adversary is not possible to decrypt the index file without a secret key. If the adversary has the secret key, they cannot able to break the cipher text because only one part of cipher text is available.

We focus on three thread models that are used in previous associated scheme. Ciphertext-only attack (COA) model: In COA model, the encrypted documents, protected index file and submitted trapdoors are allowed to access from the cloud service provider. The cloud server is continuously accessible to the secure indexes and encrypted files but the trapdoors are accessible only after submitting the search query from the data users. Known-keyword attack (KKA) model: In KKA model, the cloud service provider observes the presence of the keyword in encrypted files based on the trapdoor pairs. The trapdoors are used to retrieve the encrypted files that contain the query keyword. This model performs based on the earlier trapdoor generated by the search query keywords but its having very limited

trial of trapdoor pairs. Chosen-keyword attack (CKA) model: In CKA model, the adversary collects the information by gaining the plaintext of chosen encrypted data. The adversary has a possibility to submit one or more identified ciphertexts into the scheme and achieve t get the plaintext. The goal of the opponent is to retrieve the secure index.

The scheme is secured from above three thread models because the index and files are encrypted by AES algorithm. The symmetric encryption is well protected from COA. the keyword is split into two tokens, the first part contains starting four letters and second part contains the remaining letters. Both the tokens encrypted separately and index file has file id, R_i, n-sf, ER, frequency. When the user submit the search query, the query keyword split into two tokens and R_i, n-sf is used to search the keyword over index file. R_i, is used to select the bucket and n-sf is used select the length of the remaining word ER. The trapdoor generated based on the R_i and the adversary no idea about ER. The KKA and CKA is not possible to attack on SKFS scheme because the adversary can only access of R_i, n-sf and it is not sufficient to gather information about the keywords or index file.

8 Conclusions and future work

In this paper, we solve the problem of multi-keyword ranked search and synonym and fuzzy based keyword search over the encrypted cloud data. The proposed SKFS scheme includes multi-keyword ranked search with enhanced accuracy and synonym-based search which supports synonym queries. When the cloud user gives the synonym of any predefined keywords as input, instead of exact matching keywords, it also returns the search result for synonym queries. The inverted index method used for indexing and wildcard based technique for indexing keywords, takes less time for searching. The TF*IDF technique is used for ranking mechanism. The SKFS scheme is simulated on real-time datasets to verify the search efficiency, storage cost and security. It is observed that the SKFS scheme supports synonym and fuzzy based queries and it outperforms B-tree scheme [9].

Further, we would like to explore semantic based search over encrypted cloud data and search for images using image information.

References

1. Balamuralikrishna T, Anuradha C, Raghavendrasai N (2013) Fuzzy keyword search over encrypted data in cloud computing. Asian J Comput Sci Inform Technol 1(3)
2. Bijral S, Mukhopadhyay D (2014) Efficient fuzzy search engine with B-tree search mechanism, arXiv:1411.6773
3. Cao N, Wang C, Li M, Ren K, Lou W (2014) Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parall Distri Syst 25(1):222–233
4. ChinnaSamy R, Sujatha S (2012) An efficient semantic secure keyword based search scheme in cloud storage services. In: Proceedings of the international conference on recent trends in information technology (ICRTIT), pp 488–491
5. Chuah M, Hu W (2011) Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In: Proceedings of the 31st international conference on distributed computing systems workshops (ICDCSW), pp 273–281
6. Fu Z, Sun X, Xia Z, Zhou L, Shu J (2013) Multi-keyword ranked search supporting synonym query over encrypted data in cloud computing. In: Proceedings of the IEEE 32nd international performance computing and communications conference (IPCCC), pp 1–8
7. Fu Z, Shu J, Sun X, Linge N (2014) Smart cloud search services: verifiable keyword-based semantic search over encrypted cloud data. IEEE Trans Consum Electron 60(4):762–770

8. Fu Z, Shu J, Sun X, Zhang D (2014) Semantic keyword search based on trie over encrypted cloud data. In: Proceedings of the 2nd international workshop on security in cloud computing, pp 59–62
9. Fu Z, Sun X, Linge N, Zhou L (2014) Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. IEEE Trans Consum Electron 60(1):164–172
10. Jie W, Xiao Y, Ming Z, Yong W (2014) A novel dynamic ranked fuzzy keyword search over cloud encrypted data. In: Proceeding of the 12th international conference on dependable, autonomic and secure computing, pp 91–96
11. Khan NS, Krishna CR, Khurana A (2014) Secure ranked fuzzy multi-keyword search over outsourced encrypted cloud data. In: Proceedings of the international conference on computer and communication technology (ICCCT), pp 241–249
12. Ko J, Shin S, Eom S, Song M, Jung J, Shin DH, Lee KH, Jang Y (2014) Keyword based semantic search for mobile data. In: Proceedings of the IEEE 15th international conference on mobile data management (MDM), vol 1, pp 245–248
13. Li J, Wang Q, Wang C, Cao N, Ren K, Lou W (2010) Fuzzy keyword search over encrypted data in cloud computing. In: 2010 Proceedings IEEE INFOCOM, pp 1–5
14. Moh TS, Ho KH (2014) Efficient semantic search over encrypted data in cloud computing. In: Proceedings of the international conference on high performance computing & simulation (HPCS), pp 382–390
15. National Science Foundation Research Awards Abstracts 1990-2003. http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html (2013)
16. Raghavendra S, Geeta CM, Shaila K, Buyya R, Venugopal KR, Iyengar SS, Patnaik LM (2015) MSSS: most significant single-keyword search over encrypted cloud data. In: Proceedings of the 6th annual intrernational conference on ICT: BigData, Cloud and Security
17. Raghavendra S, Girish S, Geeta CM, Buyya R, Venugopal KR, Iyengar SS, Patnaik LM (2015) IGSK: index generation on split keyword for search over cloud data. In: 2015 international conference on computing and network communications (CoCoNet'15), pp 380–386
18. Raghavendra S, Girish S, Geeta CM, Buyya R, Venugopal KR, Iyengar SS, Patnaik LM (2015) MSIGT: Most significant index generation technique for cloud environment. In: 12th IEEE India international Conference on e^3-c^3(INDICON 2015). IEEE
19. Raghavendra S, Girish S, Geeta CM, Buyya R, Venugopal KR, Iyengar SS, Patnaik LM (2016) DRSMS: Domain and range specific multi-keyword search over encrypted cloud data. Intern J Comput Sci Inform Sec 14(5)
20. Shekokar N, Sampat K, Chandawalla C, Shah J (2015) Implementation of fuzzy keyword search over encrypted data in cloud computing. Procedia Comput Sci 45:499–505
21. Sun W, Wang B, Cao N, Li M, Lou W, Hou YT, Li H (2013) Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC symposium on information, computer and communications security, pp 71–82
22. Sun W, Wang B, Cao N, Li M, Lou W, Hou YT, Li H (2014) Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans Parall Distri Syst 25(11):3025–3035
23. Tuo H, Wenping M (2013) An effective fuzzy keyword search scheme in cloud computing. In: Proceedings of the 5th international conference on intelligent networking and collaborative systems (INCoS), pp 786–789
24. Wang B, Yu S, Lou W, Hou YT (2014) Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: Proceedings IEEE INFOCOM, pp 2112–2120
25. Wang C, Cao N, Ren K, Lou W (2012) Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parall Distri Syst 23(8):1467–1479
26. Wang C, Ren K, Yu S, Raje KM (2012) Achieving usable and Privacy-Assured similarity search over outsourced cloud data. In: Proceedings IEEE INFOCOM. IEEE, pp 451–459
27. Wang C, Wang Q, Ren K (2011) Towards secure and effective utilization over encrypted cloud data. In: Proceedings of the 31st international conference on distributed computing systems workshops (ICDCSW), pp 282–286
28. Wang D, Fu S, Xu M (2013) A privacy-preserving fuzzy keyword search scheme over encrypted cloud data. In: Proceedings of the 5th international conference on cloud computing technology and science (CloudCom), vol 1, pp 663–670
29. Wang J, Ma H, Tang Q, Li J, Zhu H, Ma S, Chen X (2013) Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. J Comput Sci Inform Syst 10(2):667–684
30. Xia Q, Ni J, Kanpogninge AJBA, Gee JC (2015) Searchable public-key encryption with data sharing in dynamic groups for mobile cloud storage. J Univ Comput Sci 21(3):440–453
31. Xia Z, Zhu Y, Sun X, Wang J (2013) A similarity search scheme over encrypted cloud images based on secure transformation. Intern J Future Gene Commun Netw 6(6):71–80

32. Xu P, Jin H, Wu Q, Wang W (2013) Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. IEEE Trans Comput 62(11):2266–2277
33. Xu Q, Shen H, Sang Y, Tian H (2013) Privacy-preserving ranked fuzzy keyword search over encrypted cloud data. In: Proceedings of the international conference on parallel and distributed computing, applications and technologies (PDCAT), pp 239–245
34. Yuan D, Yang Y, Liu X, Li W, Cui L, Xu M, Chen J (2013) A highly practical approach toward achieving minimum data sets storage cost in the cloud. IEEE Trans Parall Distri Syst (6):1234–1244
35. Zhou W, Liu L, Jing H, Zhang C, Yao S, Wang S (2013) K-gram based fuzzy keyword search over encrypted cloud computing. J Softw Eng Appl 6(1):29–32

Raghavendra S is a research scholar in the department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, India. He received his Bachelor degree in Computer Science and Engineering from BMS Institute of Technology, Visvesvaraya Technological University, Bangalore, India and Master degree from R.V.College of Engineering, Visvesvaraya Technological University, Bangalore, India. His research interests include Cloud Computing, applied cryptography and network security. He is a student member of the IEEE.

Girish S is a Master of Engineering student in the department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He completed his B.E. in Computer Science and Engineering from Sri Siddhartha Intitute of Technology, Tumkur. His areas of research interest are Cloud Computing, Data Mining and Reliability Engineering.

Geeta C. M. is a research scholar in the department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. She has received B.E.degree in Electronics and Communication, 1997 and M.E degree in Information Technology, 2007, from Bangalore University, Bangalore, Karnataka, India. Her areas of interest are cloud computing, wireless sensor networks.

Rajkumar Buyya is Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He has authored over 500+ publications and four text books including "Mastering Cloud Computing" published by McGraw Hill and Elsevier/Morgan Kaufmann, 2013 for Indian and international markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=108, g-index=225, 55200+ citations). Microsoft Academic Search Index ranked Dr. Buyya as the world's top author in distributed and parallel computing between 2007 and 2012. He is serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. He has received award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society, USA. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Asia Pacific Frost and Sullivan New Product Innovation Award" and "2011 Telstra Innovation Challenge, People's Choice Award". He served as the founding Editor-in-Chief (EiC) of IEEE Transactions on Cloud Computing?(TCC). Recently, Dr. Buyya is recognized as "2016 Web of Science Highly Cited Researcher" by Thomson Reuters. He is a Fellow of IEEE.

Venugopal K. R. is currently the Principal, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D. in Economics from Bangalore University and Ph.D. in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored and edited 65 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ and Digital Circuits and Systems etc.. He has filed 101 Patents. During his three decades of service at UVCE he has over 500+ research papers to his credit. His research interests include Computer Networks, Wireless Sensor Networks, Parallel and Distributed Systems, Digital Signal Processing and Data Mining.

S. S. Iyenger is currently Ryder Professor, Florida International University, USA. He was Roy Paul Daniels Professor and chairman of the Computer Science Department of Louisiana state University. He heads the Wireless Sensor Networks Laboratory and the Robotics Research Laboratory at USA. He has been involved with research in High Performance Algorithms, Data Structures, Sensor Fusion and Intelligent Systems, since receiving hid Ph.D degree in 1974 from MSU, USA. He is Fellow of IEEE and ACM. He has directed over 63 Ph.D students and 100 post graduate students, many of whom are faculty of Major Universities worldwide or Scientists or Engineers at National Labls/Industries around the world. He has published more than 500+ research papers and has authored/co-authored 8 books and edited 14 books. His books are published by John Wiley and Sons, CRC Press, Prentice Hall, Springer Verlang, IEEE Computer Society Press etc.. One of his books titled Introduction to Parallel Algorithms has been transalted to chinese.

L. M. Patnaik is currently Honorary Professor, Indian Institute of Science, Bangalore, India. He was a Vice Chancellor, Defense Institute of Advanced Technology, Pune, India and was a Professor since 1986 with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. During the past 35 years of his service at the Institute he has over 700 research publications in refereed International Journals and refereed International Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to High Performance Computing and Soft Computing. His areas of research interest have been Parallel and Distributed Computing, Mobile Computing, CAD, Soft Computing and Computational Neuroscience.