DRSMS: Domain and Range Specific Multi-Keyword Search over Encrypted Cloud Data

Raghavendra S*, Geeta C M*, Rajkumar Buyya†, Venugopal K R*, S S Iyengar‡ and L M Patnaik§
*Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore, India.
†Cloud Computing and Distributed Systems (CLOUDS) Lab Department of Computing and Information Systems, The University of Melbourne, Australia.
‡Department of Computer Science and Engineering, Florida International University, USA
§Adjunct Professor and INSA Senior Scientist, National Institute of Advanced Studies, Indian Institute of Science Campus, Bangalore.

***Abstract*—One of the most fundamental services of cloud computing is Cloud storage service. Huge amount of sensitive data is stored in the cloud for easy remote access and to reduce the cost of storage. It is necessary to encrypt the sensitive data before uploading to the cloud server in order to maintain privacy and security. All traditional searchable symmetric encryption (SSE) schemes enable the users to search on the entire index file. In this paper, we propose the Domain and Range Specific Multi-keyword Search $(DRSMS)$ scheme that minimizes the search time and Index storage space. This scheme adopts collection sort technique to split the index file into D Domains and R Ranges. The Domain is based on the length of the keyword; the Range splits within the domain based on the first letter of the keyword. A mathematical model is used to search over the encrypted indexed keyword that eliminates the information leakage. Binary search is used to select the range within the domain with time complexity $O(RlogD)$ and linear search is used to find the keyword within the range with $O(R)$. The space complexity of the index storage space is $O(N_T \times 3)$ and search time complexity is $O(1)+O(RlogD)+O(R)$, while the complexity of index generation is $O(N_T \times 3)$. Extensive experiments on real-world dataset validate our analysis and shows that the proposed $DRSMS$ scheme is more efficient and secure than $RSSE$ Scheme.**

***Index Terms*—Cloud Computing, Multi-keyword Search, Searchable Encryption, Data Security, $DRSMS$.**

I. Introduction

CLOUD computing has become a important deployment platform for distributed applications especially as data storage and information management service due to its enormous potential in computing, storage and various applications [1]. From a joint pool of computing resources that are configurable, it allow storage of remote data, on-demand usage. An elastic and financial plan is provided by cloud computing infrastructure for managing information and sharing resources. System maintenance overhead and hardware-software expenditure is reduced by it. It offer appropriate communication path to share resources between data owners and data users. The popularity of cloud services, such as Microsoft Azure, AWS Amazon Services, Apple iCloud, Google AppEngine, has enabled companies to shift their data onto the cloud. The data owner can deploy the personal information onto public cloud and data user can access the information anytime and anywhere. Particularly, huge amount of information and workloads can be deployed by end-user to the cloud. Usage of unlimited computing in a pay-per-use resource sharing model serve a one of the benefits and this permits the user to pay only for the amount of service used.

The highly challenge tasks faced by Cloud Computing infrastructure, data confidentiality, reliability and safety concerns occupy the main position. In practical, the public cloud which are away from the trusted domains contain the confidential data. The data uploaded by the data owners to the cloud bring concern of possible data loss, dishonest utilization of confidential data as the owners do not possess any direct control over the sensitive information. Generally, cloud servers are labelled as curious and untrusted entities. Data owner hinder to implement cloud technologies when a case of breach of information to third party or cloud provider itself is possible. Hence providing ample security and confidentiality protection to information that is susceptible to breach is of high importance. This gets employed in application designed for healthcare, financial and government data. So as to prevent the breach of more confidential data that is uploaded to the cloud, information is encrypted beforehand and then uploaded to the cloud server. To retrieve data files, traditional searchable symmetric encryption (SSE)[2] technique depends on keyword search mechanism but they support only Boolean keyword search without any assurance of n file retrieval accuracy. This mechanism is inefficient in retrieval; it demands a large amount of post-processing overhead and incurs unnecessary network traffic.

To resolves the problems of data breach in cloud, current solutions use the following approaches to provide searching ability on cloud data on basis of keywords[3]–[6]. A collection of keywords are identified and stored on the index file. For every file an index vector is designed. After the creation of index vectors, all the index vector are merged in an index file and produced. The index file thus produced and the data file are uploaded to cloud servers after encryption. Now, the information is prepared to allow queries from the datauser.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Cipher-text is supported by cloud servers based n queries as follows. A keyword based search query on the cloud containing the encrypted data is sent by the data user and keywords that are encrypted are sent to the cloud. After receiving the query, the cloud server implements as search on the encrypted index and returns a results of the list of relatable files. The data user then makes a choice of the files that are necessary and are retrieved from the cloud server. With the help of the authorized secret key, the user decrypts the required encrypted files that were retrieved from the cloud. This way, protection of data from breach and data confidentiality is safeguarded. During the entire procedure, plaintext information or keywords are invisible to the cloud servers.

*Motivation:*In the previous schemes, single keyword and multi-keyword search are used to search query over encrypted cloud data. The major obstacles in achieving these search schemes are: How to perform resourceful and secure search over encrypted data. In the previous schemes database-centre does not give supportable protection over encrypted cloud data. Current results over encrypted cloud data support only for linear search. However, given enormous amount of outsourced data, linear search is inefficient for huge data. This paper fouses on secure searching technique with resourceful and flexible search over encrypted data. We propose a new scheme that incorporates Domain and Range concept which depends on the length and starting letter of the keyword.

Contribution: In this paper, we propose a novel Index generation and a queried keyword search technique *Domain and Range Specific Multi-keyword Search* $(DRSMS)$, that supports accurate search over encrypted cloud data. $DRSMS$ provides secure, efficient and effective search results within a short time and it protects confidentially of data from the cloud service provider and unauthorised users. $DRSMS$ scheme reduces Index Storage Space by arranging keywords in an array format discussed in section VI. Specifically, our **contribution** summarised as follows:

1) We proposed a state-of-the-art information retrieval technique Domain and Range Specific Multi-keyword Search $(DRMSM)$ scheme that supports accurate and minimum search time over a large dataset.
2) The algorithm reduces index storage space and searchable time for top-k multi keyword retrieval on cloud sensitive information.
3) The time complexity of $DRSMS$ scheme is reduced to $O(N_T \times 3)$ for index building. The index storage space is of the order $O(1)+O(RlogD)+O(R)$ over encrypted cloud data.
4) A mathematical model is developed that provides security. The proposed scheme prevents sensitive information leakage thus achieving better privacy of keywords.
5) Extensive experimental evaluation demonstrates the efficiency and effectiveness of $DRSMS$.

Organisation: The rest of the paper is structured as follows: First, Literature survey is reviewed in Section 2. Ranked Searchable Symmetric Encryption $(RSSE)$ scheme and its drawbacks are discussed in Section 3. System model and design goals are defined in Section 4. Section 5 gives the detailed description of our domain and range specified search scheme. Performance analysis discussed in Section 6. Conclusion are presented in Section 7.

II. Related work

We discuss a collection of state-of-art techniques research works focused on secure ranked multi- keyword search over encrypted cloud data. We also identified their strength and limitation of the existing works.

Multi-keyword ranked search scheme have been investigated in [7]–[11]. A general framework proposed for multi-user noisy-keyword-based searchable symmetric encryption in a fault-tolerant manner [11]. Existing efforts on multi-user searchable symmetric encryption (SSE) have focused on exact keyword search, but these results are not applied to the situation where the keywords associated with the files are noisy data. It combines a single-user noisy-keyword-based SSE scheme with a private-key dynamic broadcast encryption scheme. This scheme permits dataowner to efficiently and dynamically revoke the users. Chen et al., [10] developed an Efficient and secure Semantic Multi-Keyword Ranked Search over Encrypted Cloud data. Latent Semantic Analysis (LSA) is used to reveal the relationship between terms and documents. This scheme utilize k-Nearest Neighbor $(k-NN)$ and returns the files containing the terms semantically related to the query keyword. The experimental results of LSA are better than Multi-keyword Ranked Search over Encrypted Cloud Data$(MSRE)$ scheme. The matrix index file utilises large storage space compared to other schemes.

Li et al.,[7] have designed a well-organised multi-keyword ranked retrieval scheme with Johnson-Lindenstrauss (JL) transform over encrypted cloud data. The search technique having problem of low accuracy by directly using JL transform is overcome with Optimized Maximum Query method to build an efficient trapdoor. This scheme significantly reduces the space complexity but has computation overhead. Zhang et al., [9] addressed the issue of secure ranked multi-keyword search for multiple data owners and multiple data users in the cloud computing environment. The scheme enables authorised data users to achieve protected, convenient and efficient search over multiple data owner's data that is encrypted with different secret keys to rank the search results and preserve the privacy of relevance scores between keywords and files. A new Additive Order and Privacy Preserving Function family is proposed. This scheme supports large scale datasets. Additional computation and storage cost is the overhead.

Privacy preserving keyword search schemes are proposed in [12]–[16] focusing on security encryption techniques. Li et al., [12] have designed a scalable framework for Authorised Private Keyword Search $(APKS)$ over encrypted data based on Hierarchical Predicate Encryption (HPE). In

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

this framework, every user obtained searching capabilities authorisation from Local Trusted Authority (LTA). $AKPS$ enabled multi-keyword search, allows delegation and revocation of search capabilities. The major disadvantage is that $APKS$ does not prevent keyword attack. Buyrukbilen et al., [13] designed a Privacy-Preserving Ranked Search on Public-Key Encrypted Data. This scheme employs a sample indexing structure, homomorphic encryption and private information retrieval protocols to process queries in a privacy-preserving manner. The query response time reduces by several orders of magnitude but has storage overhead and increased computation cost. Wang et al., [14] have integrated several innovative schemes to solve Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. The fuzzy multi-keyword search built the file index using LSH function in the Bloom filter technique. It gives a well-organised solution to the secure fuzzy keyword search. The Euclidean distance is implemented to capture the similarity between the keywords to calculate the similarity score to enable ranked result. It incurs computation and storage overhead.

Some of the Existing multi keyword search research works explored in [17]–[27]. Lu et al., [24] designed a novel cryptographic primitive - range predicate encryption - to build a Logarithmic Search over Encrypted Data $(LSED)$ system. This scheme is provably secure with regard to plaintext confidentiality, predicate privacy and supports logarithmic search over encrypted data, query authentication and secure data update. The $LSED$ system reveals the access patterns of cipher texts to the cloud server. Moreover, all database update operation and query authorization relies on the database owner i.e., a single point of failure. Orencik et al., [23] developed a scheme based on Public Information Retrieval (PIR) that permits multi-keyword queries with ranking facility. Symmetric- key encryption method is used for file encryption rather than public-key encryption. An efficient ranking approach based on term frequency of keywords is utilized that returns highly relevant documents corresponding to submitted search words. This scheme increases the efficiency with the help of the blinded encryption technique in accessing the contents of the retrieved documents without leaking them to other parties.
Wang et al., [18] established Static Index (SI) and Dynamic Index (DI) for Public-key Encryption with Keyword Search $(PEKS)$ to make search secure and efficient. SI and DI help $PEKS$ to decrease the load respectively in two parts: If data users are searching queried keyword for first time, SI is used or else, DI is used, SI and DI are concurrently functional with $PEKS$ and enhanced as Secure Hybrid Indexed Search $(SHIS)$ scheme that uses deterministic encryption (DE). $SHIS$ is improved further for multiple-receiver applications but this extension, supports only one keyword searchable ciphertext. Wang et al., [19] proposed a novel Fuzzy Keyword Search Scheme $(F2SE)$ that uses fingerprint extraction and secure kNN encryption algorithm to achieve a top-k ranked fuzzy keyword search. It has a low storage overhead and practical searching time cost. The fingerprint extraction algorithm can be optimized to improve Searching Accuracy Rate and match it with other symbols or languages.

Gu et al., [20] proposed Public Key Encryption with Keyword Search $(PEKS)$ scheme using lattices. $PEKS$ is a method for searching on encrypted data. It enables the user to send a secret value T_w to a server. It enables the server to place all encrypted messages containing the keyword, but without learning anything, with probabilistic consistency. The scheme is secure with the hardness of the standard Learning With Errors (LWE). The scheme focuses on security but not on computation cost. Goldreich et al., [26] have proposed oblivious RAM that uses Square-root algorithm and hierarchical solution. $RAMs$ allowed clients to completely hide the data access patterns from the cloud server provider. It can be used in conjunction with encryption to enable stronger privacy guarantees. However, utilising oblivious RAM usually brings exponential number of interactions between the user and the server for each search request.
Xia et al., [21] proposed a scheme for basic similarity search over encrypted images based on a secure transformation method that protected the information about features, and did not degrade the result accuracy. The proposed scheme protected the confidentiality of image database, feature vectors, and user's query. Moreover, the image owner could update the encrypted image database as well as the secure index quite easily. This scheme assured the confidentiality of the data, result accuracy and query unlinkability. The time complexity of query on invert index is $O(n)$, which can be further enhanced by using better indexing technique to reduce search time. Kuzu et al., [22] have proposed an efficient scheme for similarity search over encrypted data. The Locality Sensitive Hashing (LSH) algorithm is used for fast near neighbor search in high dimensional spaces. LSH provides fast similarity search in the environment of encrypted data. The experimental datasets tested on large dataset.

III. Background Work

Wang et al., [28] designed a statistical measure approach known as Ranked Searchable Symmetric Encryption(RSSE). RSSE scheme introduced Information Retrieval and text mining to embed the weight information i.e., relevance score of each file. RSSE scheme generates searchable index before outsourcing the encrypted file by using inverted index. RSSE scheme adopts one-to-many Order-Preserving mapping technique integrated with crypto primitive and Order-Preserving Symmetric Encryption $(OPSE)$ for security.

Inverted index has storage overhead (50% - 150%) on large scale datasets and high maintenance costs on updates, insertions and deletions. The processing cost increases with the number of weights in the $(m \times n)$ matrix is shown in Figure 2. Even though a term is not contained in the document, it still

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

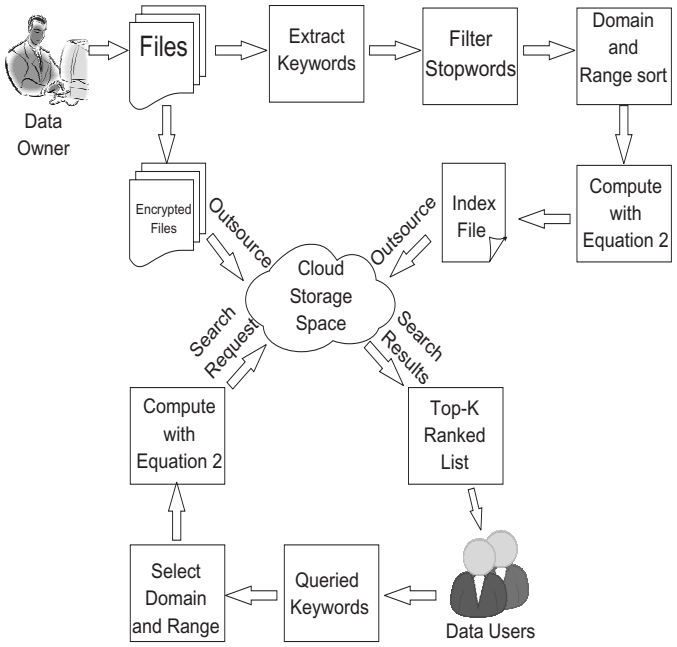

Fig. 1. System Architecture

allocates memory to store term frequency zero on the matrix. Large number of zero's appears on the index that increases the search and computation time. In $RSSE$, the analysis is performed on single keyword (w_i). The $OPSE$ technique allocates extra memory to store cipher text of each relevance score. Inverted index is not compatible for large datasets. The $RSSE$ Order-Preserving Mapping (OPM) is achieved on single keyword search with the support of Domain and Range over encrypted cloud data but not on multi keyword search. Here the Domain is distribution of relevance score for keyword and Range is $(OPSE)$ distribution score for the similar keyword.

IV. Problem Definition and System Model

A. Problem Definition

Given that n files are encrypted and uploaded onto the cloud. The main objectives are:

- To reduce the index storage space.
- To reduce search time over encrypted cloud data.
- To provides security and privacy without learning any extra information from the attackers

B. System Model

The system model is shown in Figure 1 having three entities: *Data Owner*, *Cloud Server* and *Data Users*.
The data owner is a collection of n files represented by $F = (f_1, f_2, \ldots, f_n)$ to be outsourced on the cloud space. The terms are extracted before outsourcing a file and an index file is built. The index file contains: *term, file* $ID(f_i)$ and *frequency* in the form of domain and range. It is easy to find the keyword over encrypted index file. Further the index file and collection of n files are encrypted before outsourcing to the cloud.

The cloud server communicates with the data owner and data user. It hosts storage and retrieval services for the third party. The cloud provider is not involved in any deletion or modification of data. In most of the SSE schemes, the cloud server is considered as honest-but-curious, status to learn information from stored data.
The secret key is used to generate the trapdoor t_w between the *Data User* and *Cloud Server*. The authorised user can send queried keyword to the cloud server to search the top-k files. The queried keyword search depends on domain and range of the index. After receiving the search keyword, the cloud server returns the relevant files as fast as possible related to the queried keyword. The datauser can reduce the communication cost by sending the optimal value k. The top-k results are returned to the datauser from the cloud server.

C. Design Goals

The multi-keyword ranked search can be made resourceful and safe over outsourced encrypted cloud data, only when the system concurrently accomplishes the following design goals:
Domain and Range Keyword Search: To design a domain and range multi-keyword ranked search over encrypted data which provides accurate and efficient search on document collection.
Search Efficiency: The index structure aims to improve search efficiency by exploring a domain and range based keyword search than linear search [29]
Storage Cost: To reduce index storage space compare to existing OPM scheme[28].

V. Mathematical Model

A. Score Calculation Method

The Score S is computed by calculating occurence of individual term in each file. The expression for standardized Score estimation is as per the following:

$$S = \frac{freq}{max_f req}. \tag{1}$$

where $freq$ - recurrence of each term in a record, $max_f req$ - most extreme recurrence in the wake of considering each documents in the folder and S - is Score acquired by $\frac{freq}{max_f req}$.

Another scientific model for encrypting the keyword is given below:

$$\alpha(w_i) = (a_0 x^k + a_1 x^{k-1} + \ldots\ldots + a_n x^{k-n}) \tag{2}$$

$$\alpha(w_i) = \sum_{p=0}^{n} a_b x^{k-p} \tag{3}$$

where x is a real number, k represents the length of the keyword and p is the postion of the letter in a keyword. For example, if the keyword is $sciences$ then the length is 8 and position of letter e is 4.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

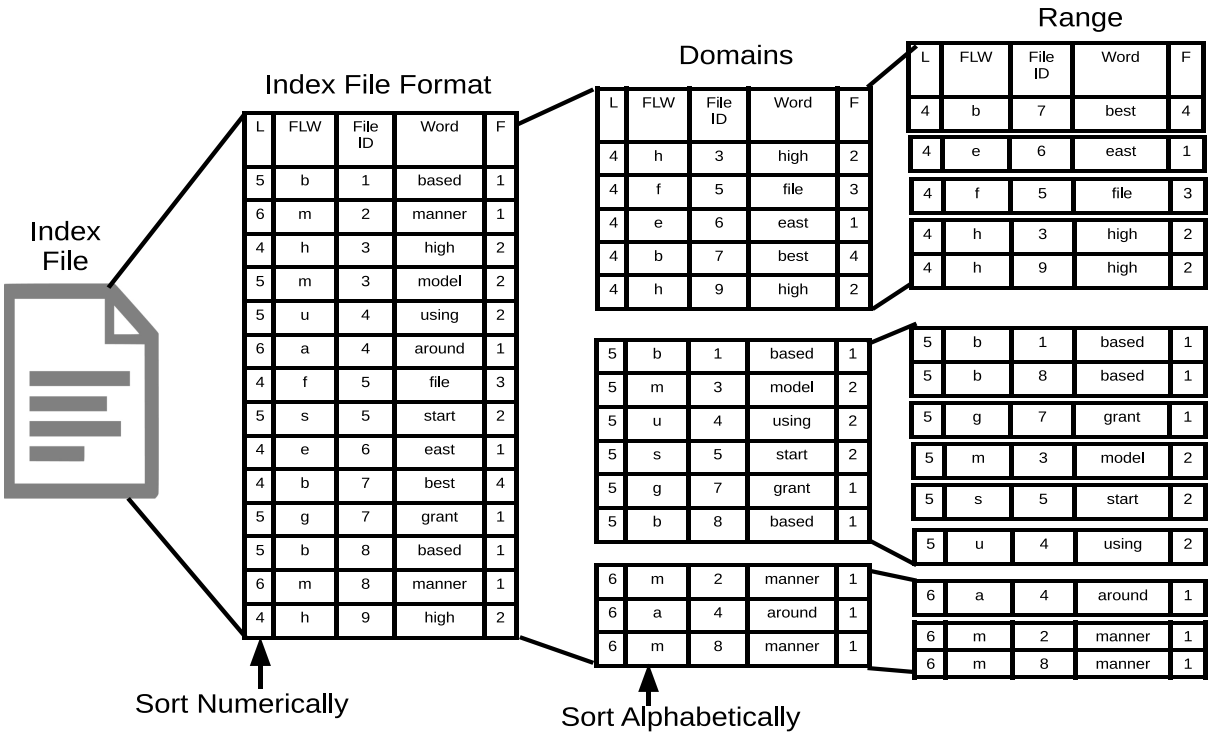

Fig. 2. Numerical Sort Splits the Index into Domains and Alphabetical Sort Splits each Domain into Ranges

B. Example

The example for the keyword $sciences$ is explained here. The ASCII (256 counts) value of individual letters are obtained; The length of the keyword is $7(length - 1)$, the constant is assumed as 2 and n starts from $0(0 \leq i \leq 7)$; the computation process is shown below for $sciences$ and the final result is $27,625$.

$$\begin{aligned}\alpha_x(w_{ij}) &= (a_0x^k + a_1x^{k-1} + a_2x^{k-2} + a_3x^{k-3} + \\ &\quad a_4x^{k-4} + a_5x^{k-5} + a_6x^{k-6}) \\ &= s \times 2^7 + c \times 2^6 + i \times 2^5 + e \times 2^4 + \\ &\quad n \times 2^3 + c \times 2^2 + e \times 2^1 + s \times 2^0 \\ &= 115 \times 2^7 + 99 \times 2^6 + 105 \times 2^5 + 101 \times 2^4 + \\ &\quad 110 \times 2^3 + 99 \times 2^2 + 101 \times 2^1 + 115 \times 2^0 \\ &= 27,625\end{aligned}$$

C. Domain and Range Sort Process

Figure 2 shows the index sort process based on Domain D and Range R. The index file $(n \times 5)$ matrix format includes $\langle L||FLW||ID(f_i)||w_i||S\rangle$. The collection sort is used for array elements in the index. The index is split into Domains D_n, that depends on the length of the word and Range R_m depends on the first letter of the word sort within the Domain D_i. Numerical sort splits the index into domains and alphabetical sort splits each Domain into Range as shown in Figure 2. The time complexity of the collection sort is $O(nlog(n))$. The summation of all Domains is equal to the index file I according to Equation 4.

$$I'' = \sum_{i=1}^{n} D_i \tag{4}$$

In Equation 1 each Domain is divided into 26 Ranges i.e., Alphabet A to Z according to Equation 5. Each Range is organized as a Bucket according to Equation 6.

$$D_i = \sum_{i=a}^{z} R_{ij} \tag{5}$$

$$R_{ij} = B_{ij} \tag{6}$$

Index I" includes the summation of Domain D_n and the summation of Range R_m is represented in the Bucket format according to Equation 7.

$$I'' = \sum_{i=3}^{n} \sum_{i=a}^{z} B_{ij} \tag{7}$$

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

TABLE I
NOTATIONS

Symbols	*Definition*
F	The collection of plain-text file are outsourced as a set of n information documents $F = (f_1, f_2, f_3, \ldots, f_n)$.
W	Distinct keywords extracted from the file collection F, is a set of m keywords $W = (w_1, w_2, \ldots, w_m)$.
I	The collection of F files generates searchable index, denoted as $(I_1, I_2, \ldots, I_m)$ where each sub-index I_i built from F_i.
t_w	The trapdoor generated by a user for search request of keyword W.
ID_{list}	The queried keyword w_i presents in a set of ranked identifiers in F files.
$ID(f_i)$	The file identifier in F_i which helps to locate the actual file.
Q	User interested Queried multi-keyword.
q_i	Individual queried keyword.
S	Score is computed by term frequency TF.
a	ASCII value of each letter in the keyword.
$\alpha(w_i)$	Extracted keyword computation results by using Equation-2.
$\alpha(q_i)$	Queried keyword computation results by using Equation-2.
D	Number of files.
T	Total terms in each file.
N_T	Total rows in index document.
C	Total columns in index document (i.e., C=5).
L	Same length of the word is grouped as Domain D_i.
FLW	Same First Letter of the Word is grouped as Range R_j with in the Domain D_i.
w_i	Individual extracted term.
F'	Encrypted n documents.
I'	Computed $\alpha(w_i)$ is stored in the Index document.
I''	Encrypted I'
C_s	Bucket start position
C_e	Bucket end position

By using above equations 4 - 6, we can prove equation 7.

$$
\begin{aligned}
Proof: I'' &= \sum_{i=0}^{n} D_n \\
&= D_3 + D_4 + \ldots + D_n \\
&= \sum_{j=a}^{z} R_3 + \sum_{j=a}^{z} R_4 + \ldots + \sum_{j=a}^{z} R_n \\
&= (B_{3a} + B_{3b} + \ldots + B_{3z}) + \\
&\quad (B_{4a} + B_{4b} + \ldots + B_{4z}) + \\
&\quad \vdots \\
&\quad +(B_{na} + B_{nb} + \ldots + B_{nz}) \\
&= \sum_{j=a}^{z} B_{3j} + \sum_{j=a}^{z} B_{4j} + \ldots + \sum_{j=a}^{z} B_{nj} \\
I'' &= \sum_{i=3}^{n} \sum_{j=a}^{z} B_{ij}
\end{aligned}
$$

VI. PROPOSED SCHEME

In this paper, we consider Domain D and Range R specific secure keyword search over encrypted cloud data for the outsourced text data. In this setting, the data owner does not have sufficient resources to store the confidential data to the semi trusted cloud server. Cloud server maintains the ability to search without revealing anything from the outsourced data except from the search and access pattern. We generate secure searchable index by using features of these sensitive information. The authorised data user can perform search on the encrypted cloud data by utilising searchable index file which returns the matched files related to the queried keyword. During these process the cloud server does not learn anything from the encrypted stored data. The data user decrypts the top-k selected document using decryption key shared by the data owner.

The DRSMS scheme is formulated as follows. Let C be the set of confidential documents and W be the set of keywords of f_i belonging to F. There are four functions namely: $Setup$, $IndexGeneration$, $QueryGeneration$ and $Search$.

- $Setup(\lambda)$: The parameter λ generates Secret Key (SK) and Public Key (PK) for the proposed scheme. Data owner distributes the secret key to the authorised users.
- $IndexGeneration(PK, F)$: From the collection of sensitive documents F, each f_i file extracts the unique keyword to construct the searchable secure index I'' via the encryption key (SK). Sorting is based on Domain D and Range R format; here Domain D is taken a length of the keyword and Range R is selected subset within the Domain D. The searchable index computes the keywords according to Equation 2 and also contains the frequency and file IDs.
- $TrapdoorGeneration(PK, REQ)$: The queried keywords request REQ generates secure trapdoor between the data user and the cloud server. The bucket trapdoor t_w is built from user's keyword request REQ and then encrypted into a secure trapdoor t_w with the public key (PK).
- $Search(I, Q)$: The queried keyword Q is computed according to the Equation 2 and compared with secure the searchable index I and returns the encrypted form top-k matching files f_i.

The frame work that is split into two parts are *Initialisation phase* and *Retrieval phase*. The initialisation phase involves the functions $Setup(\lambda)$ and $IndexGeneration(K, C)$. The function $Setup(\lambda)$ generates the keys SK and PK for communcation among the data owner, data user and the remote cloud server. The index generation function involves operation on the plaintext and it extract words from the set of plaintext documents C and generates a secure searchable index I from the extracted words. The searchable Index is a $(n \times 3)$ matrix that involves file IDs, word w_i and frequency S of w_i for convenient retrieval of data (see Algorithm 1). Most of the work process on data owner side for security reason. The detail description of function build index is given below.

Initialisation Phase:

- The data owner initiates the DRSMS scheme by calling the function $Setup(\lambda)$ to generate the Secret Key(SK) and the Public Key(PK). The authorised data users access cloud data files using the secret key provided by data owner.
- The data owner calls the function $IndexGeneration$ to

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

build index. The algorithm build index scans sensitive set of files F and then extracts n distinct keywords set, $W = (w_1, w_2, w_3, \ldots, w_n,)$ for each file f_i. After extracting a set of distinct keywords $w_i = (w_i | 1 \leq j \leq n)$, the stopwords are removed. The normalisation is computed for each file frequency S according to Equation 1. The data owner stores the extracted keywords in the index file I. Index file I stores parameters $\langle L||FLW||ID(f_i)||w_i||S\rangle$ in a $(n \times 5)$ matrix format. The $(n \times 5)$ matrix sorted according to Domain D and Range R by using a collection sort as shown in figure 3. Domain D_n sort is based on the length L of each keyword and after sorting stores $(n \times 4)$ except the L column. Each domain start and end positions are stored in the index I for reference of Domains. The domain D_i is split into range R_{ij} based on alphabetical FLW and is stored in a specific bucket B_{ij}. Each bucket range R_{ij} start and end positions are stored in the index I for reference of different Ranges.

- The index file I stores the reference of Domains, reference of Ranges and $\langle ID(f_i)||\alpha(w_i)||S\rangle$ in a $(n \times 3)$ matrix. After completing the process of Domain and Range, then $\alpha(w_i)$ is computed for each keyword w_i in the index file I and values are stored in I' according to equation 2. The computed $\alpha(w_i)$ is stored with File $ID(f_i)$ and frequency score S. Now the searchable index file I' is partially encrypted.
- The data owner encrypts both the searchable index file I' into I'' and the collection of sensitive files $F = (f_1, f_2, \ldots, f_n)$ into $F' = (f'_1, f'_2, \ldots, f'_n)$ using cryptology techniques. The encrypted searchable index I'' and encrypted files $F' = (f'_1, f'_2, \ldots, f'_n)$ are uploaded to the cloud server.

Retrieval Phase: The framework of retrieval phase has two major parts, $TrapdoorGeneration$ and $Search$. The function $TrapdoorGeneration$ generates secure gateway between the data users and the cloud server. The trapdoor performs secure search through Internet for queried keywords. The function $Search$ searches the queried keyword matches within the index file I' and retrieves the related file IDs list to the data user. The data user can download the file f_i from the list of f_n without leaking any information from the encrypted cloud data.

- The data user generates a secure trapdoor t_w for the corresponding set of keywords $Q = (q_1, q_2, \ldots, q_n)$ to retrieve the related files from the cloud server. The search queried keyword is taken as $\langle FLW||q_i||L\rangle$ for each word, where L is length of the keyword and FLW is the first letter of the word to find the specific bucket B_{ij} to search the file within the optimal time.
- Before the search operation is performed, the keyword should compute $\alpha(q_i)$ according to Equation 2 for security reason. The Domain is initialised to D_3 and selection of the Domain D_i depends on the length of the keyword D_n i.e., $(3 \leq I \leq n)$. If L is greater than D_{max} then Domain is not present within the index file I'' else D_i is incremented by 1 and the process is repeated. Each Domain has 26 Ranges, the Range is selected from the first letter of the word FLW and Binary search is applied on Domain D_i to select the range R_{ij}; then R_{ij} returns the start position C_s and the end position C_e to find the Queried keyword.
- The C_s of computed value $\alpha(w_i)$ and queried keyword computed value $\alpha(q_i)$ are compared. $\alpha(w_i)$ and $\alpha(q_i)$ are equal when the keyword matches, the matched files $ID(f_i)$ are returned from the cloud server. The rank of the retrieved files ids depend on the frequency score S and is listed in the descending order. The top-k files ID_{list} are returned from the index I'' for better efficiency. The user decrypts the interested files from the top-k file list.

Algorithm 1: Build Index

input : A Collection of n Data Files $F = (f_1, f_2, \ldots, f_n)$
output : Domain and Range sorted Index file I''
procedure: Build Index(K, F)

for $f_i \leftarrow 1$ **to** F **do**
  each file $f_i \in F$;
  Scan F and Extract each term in f_i, denoted as a $W = (w_1, w_2, w_3, \ldots, w_n,)$;
  Normalised and remove the stopwords from W;

for $i \leftarrow 1$ **to** W **do**
  count frequency of each word in f_i;
  store the $\langle L||FLW||ID(f_i)||w_i||S\rangle$ in I;

- Index I sort based on length L of keyword and store in specific Domain D_i;
- Each Domain D_i sort based on alphabets FLW and store in Specified Bucket B_{ij};

for $i \leftarrow 1$ **to** W **do**
  Compute $\alpha(w_i)$ for each keyword w_i in B_{ij}according to Equation 2;
  Each computed results stores $\langle ID(f_i)||\alpha(w_i)||S\rangle$ in ascending order of index I';

- I'' = encryption of Index file I';

return I;

VII. Performance Evaluation

National Science Foundation Research Award Abstracts 1990-2003 [30] is used to evaluate the $DRSIG$ Scheme. The abstracts have huge amount of unique technical keywords. The entire system is implemented in Java language. The Data owner and the data user use a windows platform with Intel(R) Core(TM)2 Duo CPU T6400 @2.00GHz, 3072 MB of RAM and commercial public cloud Amazon S3 service to store the Index file I'' and encrypted collection of files C'. We analyse the overall Index generation cost and per keyword index storage cost of DRSIG scheme. Experiments are performed on Index construction, score calculation and keyword search time over encrypted cloud data.

A. Storage Cost

As shown in Figure 3, the $DRSMS$ occupies less index storage space than $RSSE$ [28] scheme. $DRSMS$ scheme

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Algorithm 2: Search Query

input : Queried Keywords Q
output : Top-k Search Result ID_{list}
procedure: Search Query(K, Q)

- Search keyword taken as $\langle FLW||q_i||L\rangle$ for each word;
- Compute $\alpha(q_i)$ for each Queried keyword q_i according to Equation 2;
- D_i is initialised to 3;

while $L = D_i$ **do**
Select the Domain D_i depends on the length of the keyword out of D_n;
if $L > D_{max}$ **then**
length not found
else
D_i++

procedure: BinarySearch$(D[\,], FLW, D_s, D_e)$
return range R_{ij} select with in the Domain D_i based on the first character of the search keyword out of R_{nm};
for $i \leftarrow C_s$ **to** C_e **do**
if $(\alpha(q_i) = \alpha(w_i))$ **then**
Retrieve the file $ID(f_i)$;
else
No match found;
for $i \leftarrow 1$ **to** $ID(f_n)$ **do**
retrieve the Score S for each file $ID(f_i)$;

- Sort the Score S list in descending order;
- Retrieve the top-k ID_{list} from the Index;
- Decrypt the User interested file by using Top-k list;

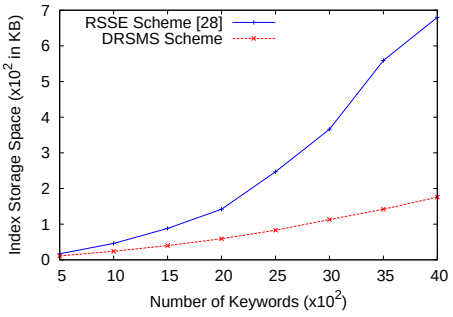

Fig. 3. Comparison of Index Storage Space based on Number of Keywords.

supports a large dataset with a minimum index storage space. The space complexity of the $RSSE$ scheme is $O(T \times D)$; here terms T are taken as maximum unique technical keywords from D files. The $RSSE$ scheme takes more storage space (i.e., 2^{30} to 2^{60}) compared to $DRSMS$ algorithm. If term T is taken as 500 distinct keywords and D is taken as 5 files. The total storage space required by $RSSE$ is 17 kilobytes i.e., $T \times D$ = 500×5 = 2500 elements, while for the same set of files, $DRSMS$ scheme requires 11 kilobytes for 5 files. Therefore, N_T = 660 and C = 3 $N_T \times C$ = 660×3 = 1980 elements. The space complexity of $DRSMS$ scheme is $O(N_T \times C)$. From numerical analysis, we observe

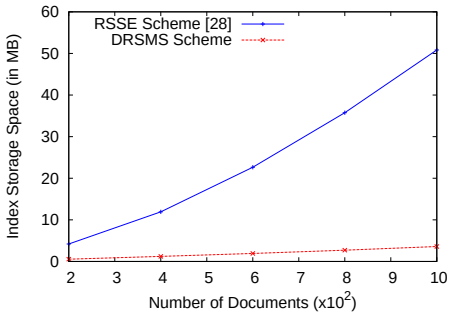

Fig. 4. Comparison of Index Storage Space based on Number of Documents.

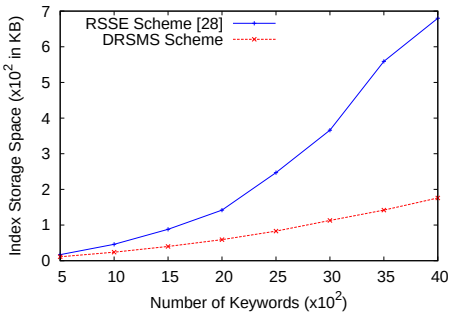

Fig. 5. Comparison of Index Generation Time based on Number of Keywords.

that $DRSMS$ scheme uses less storage space than $RSSE$ scheme. Storage space for different set of keywords is shown in Figure 3. Storage space increases exponentially with the increases in number of keywords. Experimental analysis is performed for maximum of 20,387 distinct keywords.

Figure 4 shows the index storage space for large scale of data with maximum of 1000 files, D and T is assumed as 20387 extracted distinct words from 1000 files; therefore, $T \times D$ = $1000 \times 20387 \approx$ 20.4 million elements storage space in $RSSE$ scheme. Our $DRSMS$ works on same set of 1000 files with N_T = 311800 rows of the index file organised in an array format and C = 3; therefore, $N_T \times C$ = 311800×3 = 0.9 million elements index storage space is used for 1000 files. The above Numerical analysis shows that, $DRSMS$ scheme uses less index storage space of 0.9 million elements compared with $RSSE$ scheme that uses 20.4 million elements index storage space for the same dataset. Index storage space performance improves by 35.3% for 5 files and 99.3% for 1000 files.

B. Indexing cost

The index generation time for $RSSE$ and $DRSMS$ scheme is shown in Figure 5 and Figure 6 respectively. The time complexity of index generation in $RSSE$ scheme is $O(T \times D)$. If D=5 and T=500, then $T \times D$ = 500×5 = 2500 elements; it takes 1174 milliseconds to generate index file for $RSSE$ scheme. The time complexity of index generation in $DRSMS$ scheme is $O(N_T \times C)$. If N_T=660 and C=5 (i.e., $\langle L||FLW||id(f_i)||W_{ij}||F_{ij}\rangle$), then $N_T \times C$ = 660×5 = 3300 elements; it takes 708 milliseconds to generate index file for

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

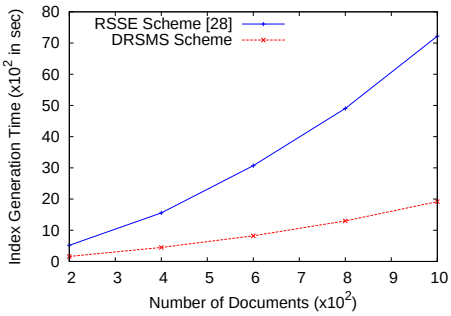

Fig. 6. Comparison of Index Generation Time based on Number of Documents.

TABLE II
SCORE CALCULATION TIME

Number of Keywords	RSSE [28] (in ms)	DRSMS (in ms)
500	103	3.4
1000	170	9.4
1500	294	20
2000	380	23.2
2500	643	31.8
3000	1092	48
3500	1399	61
4000	1489	75

$DRSMS$ scheme. $RSSE$ takes $O(n)$ for frequency score calculation and storage space range is between 2^{30} to 2^{80} bits. In the proposed $DRSMS$ scheme, we have emphasised on keyword security with minimum storage space; the number of elements in $DRSMS$ scheme is more than $RSSE$ scheme and therefore, index generation time of $DRSMS$ scheme is less than $RSSE$ scheme. This is due to computation required by $RSSE$ scheme to generate a large encrypted number (i.e., 2^{30} to 2^{80} bits range) in comparsion to $DRSMS$ scheme. The score calculation time for 5 files and 500 distinct keywords in $RSSE$ scheme is 103 milliseconds; while in $DRSMS$ scheme it is 3.4 milliseconds. The time complexity of the score calculation is $O(T \times D)$ for $RSSE$ scheme and $O(N_T)$ for $DRSMS$. If T=500 and D=5 then $T \times D$ = 500 $\times$ 5 = 2500 frequency elements; score calculation time is 103 milliseconds in $RSSE$ scheme. When N_T=660 keyword elements, score calculation time is 3.4 milliseconds for the $DRSMS$ scheme. It is observed that, with 2500 frequency elements of $RSSE$ scheme is reduces for 660 keyword elements in $DRSMS$ scheme for score calculation. The computation time of $DRSMS$ is reduced by 99.95% compared to $RSSE$ scheme. Table II and Table III shows score calculation for different set of distinct keywords and files. For 1000 files, D=1000 and T=20387, $T \times D$ = 20387 $\times$ 1000 = 20.4 million elements and the index generation time is 7221.4 seconds for $RSSE$ scheme. In $DRSMS$ scheme N_T=311800, C=5 then $N_T \times C$ = 311800 $\times$ 5 = 1.5 million elements and the index generation time is 1916.8 seconds. The index generation time in $DRSMS$ improves by 39.7% for 5 files and 73.47% for 1000 files.

TABLE III
SCORE CALCULATION TIME

Number of Files	RSSE [28] (in s)	DRSMS (in s)
200	8.112	0.293
400	29.200	0.441
600	66.930	0.709
800	118.156	1.276
1000	190.767	1.772

TABLE IV
QUERIED KEYWORDS SEARCH TIME BASED ON NUMBER OF FILES

No. of Files	Queried Keywords Search Time (in s)					
	Cell		Sciences		Investigator	
	RSSE [28]	DRSMS	RSSE [28]	DRSMS	RSSE [28]	DRSMS
200	3.290	0.886	1.803	0.709	2.088	0.219
400	4.936	1.928	3.000	1.709	4.250	0.448
600	9.253	3.410	5.193	2.689	7.475	0.724
800	10.952	5.247	8.144	3.690	11.073	0.981
1000	14.816	6.891	8.632	4.924	16.647	1.281

C. Searching Cost

Table IV shows queried keyword search time over number of documents. The time complexity of keyword search on encrypted cloud data is $O(T) + O(logn)$ for $RSSE$ scheme, where n is number of frequency score in encrypted form. In our proposed $DRSMS$ scheme, the time complexity is $O(1)$ for selecting the domain based on length of the keyword using hash map technique. The time complexity of $O(logn)$ for selecting the range within the domain based on the first character of the keyword using binary search, n has 26 different ranges (i.e., A-Z). Linear search is used to find the keyword within the range and hence the time complexity is $O(R)$, where R is number of encrypted keyword within the bucket. $DRSMS$ has three different cases 1) *Best case* is $O(1)$, if the range bucket has only one term. 2) *Average case* is $O(R)$, if the range bucket has B terms out of $N_T(1 < B < N_T)$. 3) *Worst case* is $O(N_T)$, if the range bucket has all the words in the index file. The proposed $DRSMS$ algorithm total keyword search time complexity is $O(1)$ for the best case and $O(R)$ for average and the worst case.

Example: If D=1000 files and T= 20387 distinct keywords therefore, $T \times D$ = 20387 $\times$ 1000 = 20.4 million elements. Search process is performed on 20,387 element out of 20.4 million elements in $RSSE$ scheme. In $RSSE$, the time taken to search the queried keyword is 14.82 seconds for $cell$, 8.63 seconds for $sciences$ and 16.65 seconds for $investigator$. Similarly, if there are 1000 files in $DRSMS$ scheme, N_T= 311800 keyword elements in the index file I". The time taken to search for queried keyword in $DRSMS$ scheme is 6.89 seconds for $cell$ and 4.92 seconds for $sciences$ and 1.28 seconds for $investigator$. User search comparison process is done on the specific bucket B_{ij}, the searching $cell$ keyword bucket contains 3104 elements, $sciences$ keyword bucket contains 2811 elements and $investigator$ keyword search

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

on 1402 elements out of 311800. In the proposed $DRSMS$ scheme, the queried multi-keyword search time performance improves by 83% for 5 files and 67.34% for 1000 files in comparsion to the $RSSE$ scheme. Table IV depicts search time for different number of files over encrypted cloud storage data. It is observed that there is a linear growth in the search time.

VIII. Conclusions

In this paper, we introduce a new multi-keyword search technique for achieving effective data utilisation through Internet over remotely stored encrypted cloud data. $RSSE$ scheme is inefficient to achieve multi-keyword ranked search on a large dataset. To overcome the drawback in $RSSE$ scheme, we propose a new Domain and Range Specific Multi-keyword Search $(DRSMS)$ algorithm that supports more efficient and accurate search. A mathematical model provides secure search on index file without leakage of information. DRSMS algorithm performs effective and efficient indexing and searching on encrypted data. The algorithm reduces index storage space and ranked searchable encryption time for top-k multi keyword retrieval on cloud sensitive information. Indexing phase reduces index computation time and index storage space up-to 90%. Domain D is a sort based on length of the keyword and Range R is sorted based on first character of the keyword. Searching phase reduces queried keyword search time up-to 62%. The $DRSMS$ scheme requires a small amount of additional time to sort on index file. As part of our future work we aim to reduce search time on the image data set.

References

[1] R. Buyya, R. N. Calheiros, J. Son, A. V. Dastjerdi, and Y. Yoon, "Software-Defined Cloud Computing: Architectural Elements and Open Challenges," *in Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1–12, 2014.

[2] Y. J. Earn, R. Alsaqour, M. Abdelhaq, and T. Abdullah, "Searchable Symmetric Encryption: Review and Evaluation." *in Journal of Theoretical & Applied Information Technology*, vol. 30, no. 1, 2011.

[3] S. Raghavendra, C. M. Geeta, K. Shaila, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, ""MSSS: Most Significant Single-keyword Search over Encrypted Cloud Data"," *in Proceedings of the 6th Annual Intrernational Conference on ICT: BigData, Cloud and Security*, 2015.

[4] S. Raghavendra, C. M. Geeta, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "MSIGT: Most Significant Index Generation Technique for Cloud Environment," in *12th IEEE India International Conference on E^3-C^3(INDICON 2015)*. IEEE, December 17-20 2015.

[5] S. Raghavendra, S. Girish, C. M. Geeta, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "IGSK: Index Generation on Split Keyword for Search Over Cloud Data." in *2015 International Conference on Computing and Network Communications (CoCoNet'15)*. IEEE, December 16-19 2015, pp. 380–386.

[6] S. Raghavendra, C. M. Geeta, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "DRSIG: Domain and Range Specific Index Generation for Encrypted Cloud Data"," in *Proceedings of the International Conference on Computational Techniques in Information and Communication Technologies, ICCTICT 2016*. IEEE, March 17-20 2016.

[7] K. Li, W. Zhang, K. Tian, R. Liu, and N. Yu, "An Efficient Multi-keyword Ranked Retrieval Scheme with Johnson-Lindenstrauss Transform over Encrypted CloudData," *in Proceedings of the International Conference on Cloud Computing and Big Data (CloudCom-Asia)*, pp. 320–327, 2013.

[8] R. Li, Z. Xu, W. Kang, K. C. Yow, and C.-Z. Xu, "Efficient Multi-Keyword Ranked Query over Encrypted Data in Cloud Computing," *in Future Generation Computer Systems*, vol. 30, pp. 179–190, 2014.

[9] W. Zhang, S. Xiao, Y. Lin, T. Zhou, and S. Zhou, "Secure Ranked Multi-keyword Search for Multiple Data Owners in Cloud Computing," *in Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 276–286, 2014.

[10] L. Chen, X. Sun, Z. Xia, and Q. Liu, "An Efficient and Privacy-Preserving Semantic Multi-Keyword Ranked Search over Encrypted Cloud Data," *on International Journal of Security and Its Applications*, vol. 8, no. 2, pp. 323–332, 2014.

[11] X. Pang, B. Yang, M. Zhang, and H. Wang, "Multi-user Noisy Keyword Search over Encrypted Data," *in Journal of Computational Information Systems*, vol. 9, no. 5, pp. 1973–1981, 2013.

[12] M. Li, S. Yu, N. Cao, and L. Wenjing, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," *in Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS)*, pp. 383–392, 2011.

[13] S. Buyrukbilen and S. Bakiras, "Privacy-Preserving Ranked Search on Public-Key Encrypted Data," *in Proceedings of the 2013 IEEE International Conference on High Performance Computing and Communications*, pp. 165–174, 2013.

[14] 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 the 2014 International Conference on INFOCOM*, pp. 2112–2120, 2014.

[15] Q. Liu, G. Wang, and J. Wu, "Secure and Privacy Preserving Keyword Searching for Cloud Storage Services," *in Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 927–933, 2012.

[16] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private Query on Encrypted Data in Multi-user Settings," *in Proceedings of the International Conference on Information Security Practice and Experience*, pp. 71–85, 2008.

[17] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig, "Multi-dimensional Range Query over Encrypted Data," *in IEEE Symposium on Security and Privacy*, pp. 350–364, 2007.

[18] W. Wang, P. Xu, H. Li, and L. T. Yang, "Secure Hybrid-Indexed Search for High Efficiency over Keyword Searchable Ciphertexts," *Future Generation Computer Systems*, 2014.

[19] D. Wang, S. Fu, and M. Xu, "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, 2013.

[20] C. Gu, Y. Guang, Y. Zhu, and Y. Zheng, "Public Key Encryption with Keyword Search from Lattices," *in International Journal of Information Technology*, vol. 19, no. 1, pp. 1–10, 2013.

[21] Z. Xia, Y. Zhu, X. Sun, and J. Wang, "A Similarity Search Scheme over Encrypted Cloud Images based on Secure Transformation," *in International Journal of Future Generation Communication & Networking*, vol. 6, no. 6, pp. 71–80, 2013.

[22] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient Similarity Search over Encrypted Data," *in Proceedings of the 28th International Conference on Data Engineering (ICDE)*, pp. 1156–1167, 2012.

[23] C. Örencik and E. Savaş, "Efficient and Secure Ranked Multi-keyword Search on Encrypted Cloud Data," *in Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pp. 186–195, 2012.

[24] Y. Lu, "Privacy-preserving Logarithmic-time Search on Encrypted Data in Cloud." *in 19th Annual Network and Distributed System Security Symposium, (NDSS)*, 2012.

[25] E.-J. Goh, "Secure Indexes." *in IACR Cryptology ePrint Archive*, p. 216, 2003.

[26] O. Goldreich and R. Ostrovsky, "Software Protection and Simulation on Oblivious RAMs," *in Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[27] A. Boldyreva, N. Chenette, Y. Lee, and A. O'neill, "Order-Preserving Symmetric Encryption," *in Proceedings of the International Conference Advances in Cryptology-EUROCRYPT*, pp. 224–241, 2009.

[28] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced CloudData," *in IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467–1479, 2012.

[29] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," *in IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[30] "National Science Foundation Research Awards Abstracts 1990-2003," *http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html*, 2013.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500