MSIGT: Most Significant Index Generation Technique for Cloud Environment

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. e-mail: raghush86@gmail.com.
†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
§Indian Institute of Science, Bangalore, India.

***Abstract*—Cloud Computing is a computing paradigm for delivering computational power, storage and applications as services via Internet on a pay-as-you-go basis to consumers. The data owner outsources local data to the public cloud server to reduce the cost of the data management. Critical data has to be encrypted to ensure privacy before outsourcing. The state-of-the-art SSE schemes search only over encrypted data through keywords, hence they do not provide effective data utilisation for large dataset files in cloud. We propose a Most Significant Index Generation Technique(MSIGT), that supports secure and efficient index generation time using a Most Significant Digit(MSD) radix sort. MSD radix sort is simple and faster in sorting array strings. A mathematical model is developed to encrypt the indexed keywords for secure index generation without the overhead of learning from the attacker/cloud provider. It is seen that the MSIGT scheme can reduce the cost of data on owner side to $O(N_T \times 3)$ with a score calculation of $O(N_T)$. The proposed scheme is effective and efficient in comparison with the existing algorithms.**

Keywords*—Keyword Search, Radix Sort, Data Privacy, Cloud Computing.

I. Introduction

Cloud computing is a shared pool of computing resource to store or access data from a remote place and pay for the services used by consumers. Enterprises outsource their data on the cloud. On-demand resource availability and pay-as-use concept has attracted many benefits of new computing model including relief in storage management, global data access and capital expenditure avoiding on software, hardware and maintenances [1]. Most of the companies face problem in secure information storage and retrieval on cloud. Data needs to be encrypted before outsourcing to cloud containing sensitive information like financial transactions, medical records [2], [3] and government documents etc. The cloud provider and unauthorised person can access the data from the untrusted cloud. Such cases of data loss and privacy breaches in cloud computing systems are reported in [4], [5].

Companies, health care centers and government are outsourcing the documents onto the cloud storage space since they are finding it difficult to maintain the hardware infrastructure on premises. Companies like Amazon, Windows Azure, IBM etc.. provide cloud services based on IaaS(Infrastructure-as-a-Service). Data is encrypted before outsourcing for privacy concerns; the data owner shares the encrypted data with a cloud server and then it is retrieved whenever required. Effective data utilisation is a challenging task for a large number of outsourced data files. Keyword-based search is one of the most popular technique and evolutionary approach for index generation [6] [7] can be used for searching documents on encrypted cloud data. Keyword search techniques are widely used in plain-text scenarios and the user is allowed to retrieve select files from the storage space.

All traditional Searchable Symmetric Encryption (SSE) (e.g., [8], [9], [10]) schemes allow a user to search on cipher text and securely retrieve the cipher text over encrypted cloud data through keywords without decrypting the files. This supports only Boolean keyword search without considering any relevance of the document. Boolean keyword search has a main drawback whenever a huge number of documents are involved. The user wants to find matching document for each search request without the pre-knowledge about the encrypted cloud data and wants to scroll through the entire retrieved files, which requires (i) large amount of post-processing when they go through unrelated files resulting in enormous network traffic. (ii) it incurs communication overhead. The above drawbacks can be overcome with top-k single keyword retrieval techniques [11], [12] and single-keyword retrieval techniques [13], [14].

Motivation: In the previous schemes, Boolean and single-keyword search are used to search keywords on encrypted cloud data. The main issue is to reduce the cost of computation without affecting the extracted keyword of the documents to provide security and to select accurate keyword search over encrypted cloud data.

Contribution: In this paper, we have developed a new index building technique known as MSIGT Scheme, obtain additional protection to the sensitive data from Cloud Service Provider and unauthorised entities. We address these Challenges by our proposed MSIGT scheme for secure and efficient single-keyword search over encrypted cloud data.

1) The MSIGT scheme over encrypted cloud storage data reduces index generation time over large data sets on cloud.

978-1-4673-6540-6/15/$31.00

2) The MSIGT algorithm reduces the index generation time to $O(N_T \times 3)$.
3) A new mathematical model is developed for secure index contruction, without learning anything about keyword from unauthorised entities.

Organisation: The rest of this paper is organised as follows: Related works are discussed in Section 2. Background work with respect to traditional schemes is presented in section 3. Problem statement and system model are explained in section 4. In section 5, the proposed Searchable Encryption Scheme is developed. Performance Analysis is given in section 6. Conclusions are presented in section 7.

II. LITERATURE SURVEY

We discusses various state-of-the-art technique focused on secure keyword search on data storage in cloud computing environments. We also identify their strength and weakness. In rest of paper we demonstrate how our scheme overcomes those weaknesses.

Ranked searchable encryption schemes are explored in [11],[15], [16] over encrypted cloud data. The secure ranked keyword search schemes are stronger security definition compared to SSE. The Order Persevering Mapping(OPM) technique is used in [11],[15],[16] for ranking the searched file over encrypted cloud data. The OPM technique protects the sensitive score information from the cloud provider. OPM method is highly efficient but they lead to collisions in the network and increases computation cost. One-to-many OPM technique is presented in [15],[16] for secure term frequency.Sun *et al.,* [15] developed a Secure Ranked Semantic Keyword Search (RSS) over encrypted cloud data. A fuzzy solution contributes to search the semantic keyword on encrypted cloud data. The data owner generates a piece of metadata for each file and uploads the encrypted set of metadata and collection of document to the cloud. Semantic search returns exactly matched semantically related files to the queried keyword.

Ning et al., [13] Searchable encryption technique is helpful for retrieving documents from the cloud data centres. A secure k-Nearest Neighbour (kNN) technique was implemented in MRSE Scheme, in which two threat models Cipher-text Model and Background Model are investigated with respect to privacy and efficiency in multi-keyword ranked search. Orecik et al., [17] proposed an efficient privacy-preserving search over encrypted cloud data that utilises minhash functions to improve the precision rate. The advantages of this scheme are multi keyword search in a single query and effective ranking capability based on term frequency and inverse document frequency. Secure Multi-Keyword Search Method is efficient, effective and privacy-preserving but the server computation is more.

Sun et al., [18] The index tree is constructed based on vector space model to provide flexible update operations. Cosine similarity measures gives accurate ranked search result. Greedy depth-first traverse strategy algorithm and the known cipher-text threat model are used to improve search efficiency and security. Dynamic multi-keyword ranked search scheme have communication and storage overhead. Sun et al., [19] presented a tree-based index structure and multi-dimensional (MD) algorithm gives better search efficiency than linear search. The search process is verifiable in case the user wants to confirm authenticity of the returned search results. Performance is improved in terms of efficiency and privacy but computation complexity is high.

III. BACKGROUND WORK

Yu et al., [20] have developed Searchable Encryption(TRSE) technique supporting top-k multi keyword extraction from the cloud storage system. Searchable Symmetric Encryption (SSE) technique is used to retrieve encrypted data over cloud. The homomorphic encryption is implemented to rank the searched data. The TRSE technique ensures security for small datasets. The user encrypts and send the cipher-text to the cloud server. The size of the cipher-text is too large. Therefore, the encrypted trapdoor size is too large for communication. The computation overhead on server side is dependent on $tf - idf$ weights to calculate relevance score for each keyword search request.

A. Vector Space Model

The vector space model does not effectively work on large scale datasets [21]. The vector space model matrix involves keywords (w_i), file identifiers ID_i and frequency score (S). The index file takes more and unnecessary storage space. If the frequency is zero then the keyword (w_i) does not appear in the file ID_i but the memory is allocated to store the value zero which increases the search time because of a large number of zeros. In TRSE, the analysis is limited to 1000 keywords (w_i). Homomorphic encryption computes entire matrix including zeros that incurs high computation overhead.

B. Radix Sort

The radix sort is an algorithm to rearrange the string representation process over individual string either in ascending or descending order. It is a non-comparative and a linear sorting string algorithm . The string sorting algorithm sorts data by grouping keys which shares the same significant position and value. The string sorting are of two types: least significant digit (LSD) radix sort and MSD radix sort.

1) MSD: In MSMS scheme, function 1 focuses on MSD radix sort process. The string representation starts from the most significant digit and moves towards the least significant digit (left most string to right most string for each word). LSD radix sort works in different way. It is suitable for string like words (variable-length alphabets) and fixed-length integer. Counting sort for one level of buckets to group the keys is used. The indexes file I partitions into R pieces according to the first character and groups the elements with the same character into a bucket. It processes recursively sorting from left to the right of string in each bucket. Finally, all the buckets are concatenated in order.

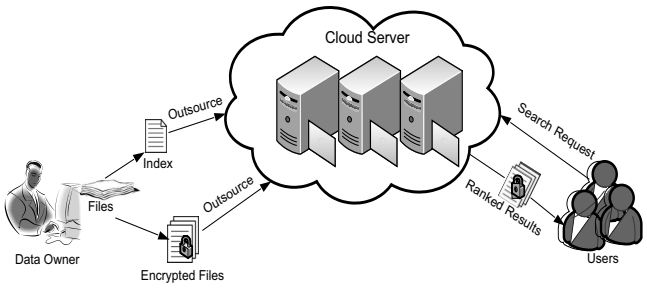

Fig. 1. System Architecture

2) Counting Sort: After MSD sorting, the string is processed with counting sort algorithm to count the objects of the bucket element [22]. An array A of n most significant elements is taken from the keywords in the range $(1, 2, \ldots, k)$. The counting sort keeps an auxiliary array $C[i]$ range 1 to k initialised to $zero$. The algorithm makes a pass through input array A; for each element i of A, increments $C[i]$ by 1. The iteration is done for n elements of array A with time complexity of $O(n)$ times and updates C. The index j values of C shows the number of times that j appears in A. The next step is to insert each element j with a total of $C[j]$ times in the new list of C' and it computes with complexity $O(k)$. The total time taken for counting sort is of the order $O(n + k)$.

IV. PROBLEM STATEMENT AND SYSTEM MODEL

Consider n files to be outsourced on the cloud server of $honest - but - curious$ model used in most of the SSE scheme. This cloud server should act as an honest follower of the designed protocol. It should be curious enough to infer and analyse the word during message flow to learn additional information. The main objectives are to:

- reduce the secure index generation technique over encrypted cloud data without learning any extra information.
- reduce communication overhead.

A. System Model

The architecture of encrypted cloud hosting services is shown in Figure 1. It contains three entities: data owner, data user and cloud server. The System provides most significant keyword search over encrypted cloud data
A collection of n data files $F = (f_1, f_2, f_3, \ldots, f_n)$ to be outsourced onto the cloud server in encrypted form and then provide keyword search services to authorised users. The files extract the m keywords $W = (w_1, w_2, \ldots, w_n)$ from each file to generate encrypted searchable index I'' from F' and store both the index I'' and encrypted file collection F' on the cloud.

The Data User is authorised to process multi-keyword search on encrypted cloud data. After conformation of authorisation, the trapdoor t_w of the keyword w_i is generated to send the encrypted search query on the cloud server. When data user submits the search query, the cloud server is searches the index I' and returns the relevant files to the data user. The ranked criteria is used to enhance the file retrieval accuracy of the search result. The data user can reduce the communication cost by sending the optimal value k along with trapdoor t_w and the cloud server sends back the top-k search results relevant to the data user's interested keyword w_i.

TABLE I. NOTATIONS

$Symbols$	$Definition$
F	The plain-text document collection to be outsourced as a set of n data files $F = (f_1, f_2, f_3, \ldots, f_n)$.
W	The extracted distinct keywords from the document collection F, a set of m keywords $W = (w_1, w_2, \ldots, w_m)$.
I	The searchable index built from the document collection F, denoted as $(I_1, I_2, \ldots, I_m)$ where each sub-index I_i build from F_i.
t_w	The trapdoor generated for search request of keyword W.
ID_{list}	The set of ranked identifiers of files in F that contains keyword w_i.
$ID(f_i)$	The file identifiers f_i to locate the actual file.
S	Score is calculated by using term frequency TF.
a	ASCII value of alphabets in each letter of the keyword.
$\alpha(w_i)$	Compute result for each extracted keyword from equation-2.
D	Number of Documents.
T	Number of terms in each document.
N_T	Number of rows in index file.
C	Number of columns in index file (i.e., C=3).
$c(i)$	Contains elements in the sort list $(i = 1, 2, \ldots, n)$.
w_i	Individual extracted keyword.
F'	Encrypted n files.
I'	Stored all the computed $\alpha(w_i)$ in the Index
I''	Encrypted I'
$A(j)$	Numbrer of extracted keyword W in an array taken as input for Function 1.

The third party data storage and retrieval service hosts on cloud server. which is termed as honest-but-curious in our model because they are communicating with both data owners and data users. The storage data may contain sensitive data, and hence the cloud server cannot be fully entrusted in protecting data. They do not delete or modify the user data but try to learn the content of the stored data.

B. Keyword Compution Method

The Score S is calculated based on the frequency of each term in the individual file. The expression for normalised Score calculation is as follows:

$$S = \frac{freq}{maxfreq}. \tag{1}$$

where $freq$ - frequency of each term in a file, $maxfreq$ - maximum frequency after considering all the files in the folder and S - is Score obtained by $\frac{freq}{maxfreq}$. A new mathematical 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 and it should be same for both index keyword and queried keyword, k - is a length of the keyword(i.e., if the keyword is $Network$ than the length of the keyword is 7) and p - is the position of the each letter $(0 \leq p \leq n)$ (if the keyword $Network$ position of letter e is 2 and r is 6).

Function 1: MSD Radix Sort

input : Extracted Keyword from the text $W = (w_1, w_2, \ldots, w_n)$ and most significant digit d

output : Index I sort by most significant digit in Ascending Lexicographical order

Function: MSD(W, d)

1)Take the MSD for the first character of each w_i;
2)Sort the W based on the first digit of each keyword w_i using `countingsort()`;
3)Grouping elements with the same digits into a bucket $B_i.(i.e., i = 1, 2, \ldots, n)$;
4)concatenate the buckets $(B_1, B_2, B_3, \ldots, B_n)$ together in order;

Procedure `countingsort(A[j])`

```
for i ← 1 to k do
    C[i] ← 0;
for j ← 1 to n do
    C[A[j]] ← C[A[j]] + 1
;
for i ← 2 to k do
    C[i] ← C[i] + C[i − 1]
;
for j ← n downto 1 do
    B[C[A[j]]] ← A[j];
    C[A[j]] ← C[A[j]] − 1;
```

Algorithm 1: Most Significant Multi-keyword Search (MSIGT)

Initialisation Phase

input : A set of n Data Files $F = (f_1, f_2, \ldots, f_n)$

output : Index file generated from extracted keyword I

Function: BuildIndex(K, F)

for $f_i \leftarrow 1$ **to** n **do**
each file $f_i \in F$;
Scan F and Extract the distinct word in f_i, denoted as a $W = (w_1, w_2, w_3, \ldots w_n,)$;
Normalised and filter the stopwords from W;
for $j \leftarrow 1$ **to** m **do**
each file $w_i \in W$;
1) Calculate the Score S for each keyword w_i according to equation 1;
2) `MSD()` to sort the Index I;
3) Compute $\alpha(w_i)$ for each keyword w_i according to equation 2;
4) Store the $\langle id(f_i)||\alpha(w_i)||S\rangle$ as an element in the posting list of I' ;
5) Encrypt the Index file I';
6) Replace I' with I''
return I'';

V. PROPOSED MSIGT SCHEME

MSIGT scheme generates index for optimal Secure multi-keyword search time over encrypted files and reduce the index storage space in the cloud sever. The framework of MSIGT scheme involves four functions $Setup$, $BuildIndex$, $TrapdoorGen$ and $SearchResult$.

- $Setup(\lambda)$: The secure input parameter λ generates Public Key(PK) and Secret Key(SK) for the Most significant digit searchable encryption scheme. The data owner distributes Secret Key to the authorised users.
- $BuildIndex(F, PK)$: The collection of files F extracts the unique keyword to construct the searchable index I. Sorting is based on the MSD radix sort technique and computes keywords according to Equation 2 in the searchable index I'. The searchable index I' also contains frequency based relevance score and file IDs. Finally, the searchable index I' is encrypted into I'' with PK, the output I'' is uploaded to the cloud storage space.
- $TrapdoorGen(I', t_w)$:The data user generates secure trapdoor t_w corresponding to the interested query keyword request Q. the multi-keyword request encrypts into a secure trapdoor t_w to search on the encrypted data I'.
- $SearchResults(I', t_w)$: On receiving the Secure trapdoor t_w. The cloud server computes and returns the matched file IDs and relevant score gets back to the users in the descending order. The top-k matched files is sent back in a ranked sequence based on the relevance score. Top-k files are securely retrieved without learning anything about the search keyword and index I'.

The framework of the Algorithm 1 can be built in two phases: (i) initialisation phase and (ii) retrieval phase. The initialisation phase involves $Setup$ and $BuildIndex$ functions. The Setup function processes on data owner side to generate SK and PK for individual authorised users. BuildIndex function involves operations on plain-text and generates secure searchable index from the plain-text files F. The searchable index $(n \times 3)$ matrix involves extracted keyword, file IDs and score for convenient retrieval of data from the Function 1 and the initialisation phase. For security concerns, most of the work is conducted on the data owner side. The details of the $initialisation$ and $retrieval$ phase is described below:

Initialisation Phase:

- The data owner generates Secret Key(SK) and Public Key(PK) for the $MSIGT$ scheme by calling the function $KeyGen(\lambda)$. The data owner shares the secret keySK with the authorised data users to access cloud data files $f_i(1 \leq i \leq n)$.
- The data owner extracts the collection of m keywords W=$(w_1, w_2, w_3, \ldots, w_m)$ from the scanned files where $W = (w_i | 1 \leq i \leq m)$ and then filters

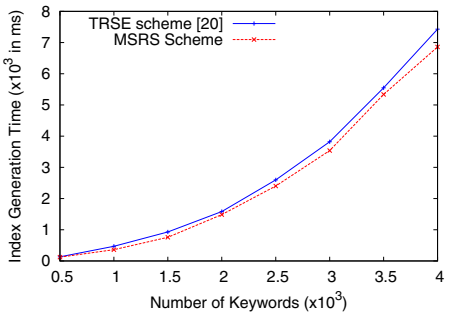

Fig. 2. Comparison of Index Generation Time based on Number of Keywords.

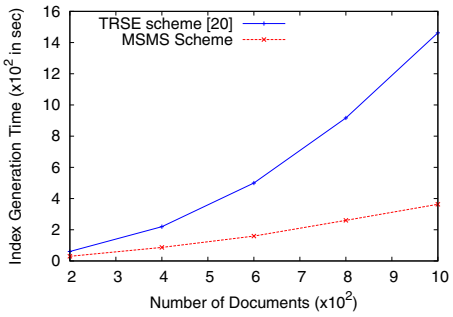

Fig. 3. Comparison of Index Generation Time over Number of Documents in Dataset.

the stop words from W and term frequency TF. Compute for each file f_i belonging to F, the score S is calculated according to Equation 1. The data owner builds a $(n \times 3)$ matrix to store the extracted keywords in the form $(w_i||ID(f_i)||S)$ into the index file I. W is the Extracted keywords after filtering the stopword, $ID(f_i)$ is the file identifiers for F files and S is a score calculated by normalised term frequency (TF). The index file I is sorted by using MSD Radix sort $Function1$. The explanation is elaborated in session 3. The counting sort function the group elements with the same MSC into a bucket $B_i(i = 1, 2, \ldots, n)$ and then concatenates the buckets $B_i = (B_1, B_2, \ldots, B_n)$ together in lexicographical order. After sorting all the extracted keyword present in the index file I, each keyword is encrypted according to Equation -2. Now the searchable index file I' is partially encrypted.

- The data owner encrypts both the Searchable index file $I' = (v'_i|1 \leq i \leq n)$, where $v'_i = (\alpha(w_i||ID(f_i)||S)$ into I'' and the file $F = (f_1, f_2, \ldots, f_n)$ into $F' = (f'_1, f'_2, \ldots, f'_n)$ with cryptology techniques and then the I'' and F' is outsourced to the cloud storage space.

VI. PERFORMANCE EVALUATION

The overall performance of our proposed scheme on the real data set: National Science Foundation Research Awards Abstracts 1990-2003 [23]. The experiment environment involves a server and a client. The entire system is conducted in Java language platform CPU E31220 @3.10GHz Quad Core processor. The index file I'' and the encrypted collection of files F' is stored on the commercial public cloud, Amazon cloud services like $S3$ (simple storage service). The storage cost of MSIGT scheme is analyzed and then experiments are performed to test index generation and score calculation time over the cloud data. The $MSIGT$ scheme is compared with $TRSE$ [20] scheme.

A. Computation Cost on Owner Side

Table II shows the experimental results of the score calculation time based on the word size and files. The time complexity of score calculation in TRSE is $O(D \times T)$, which is more than the time complexity of MSIGT scheme with $O(N_T)$. As an example TRSE scheme takes $(D \times T)$ = 1000 $\times$ 20387 $\approx$ 20 million elements, the time taken to compute 20 million elements is 1023 seconds. In MSIGT scheme, N_T $\approx$ 0.3 million elements and takes 0.091 seconds to compute the score calculation. Table II shows for different set of files in both the schemes; the number of files increases linearly with increase in time. The analysis shows that TRSE scheme takes more time to compute score calculation than MSIGT scheme.

TABLE II. SCORE CALCULATION TIME

Number of Keywords	TRSE [20] (in ms)	MSIGT (in ms)
500	12	1.6
1000	88	3.42
1500	165	4.03
2000	292	4.30
2500	509	4.75
3000	949	5.07
3500	1464	5.41
4000	2121	6.06
96247	-	1487

Number of Files	TRSE [20] (in s)	MSIGT (in s)
500	204.605	0.041
1000	1023.262	0.096
1500	2847.288	0.120
2000	5478.250	0.190
2500	-	0.220
3000	-	0.255
3500	-	0.408
4000	-	0.574
10000	-	1.482

Figure 2 and Figure 3 shows the index generation time over TRSE and MSIGT scheme. The index generation time complexity of TRSE is $O(D \times T)$. The MSIGT scheme index generation time complexity is $O(N_T \times C)$. If D=5, T=500, $D \times T$ = 5$\times$500 = 2500 elements takes 133 milliseconds to generate an index in TRSE scheme. For the same number of files, N_T=623, C=3, $N_T \times C$ = 623$\times$3 = 1869 elements takes 122 milliseconds to generate an index in MSIGT scheme. The number of documents increases linearly with increase in the index generation time. Figure 3 shows that with D=1,000 and T=20,387, $D \times T$ i.e. 1,000$\times$20387 $\approx$ 20 million elements and these elements takes 1462 seconds for index generation in TRSE scheme. For the same set of 1000 files, N_T = 2,93,842 elements, C=3, $N_T \times C$ = 2,93,842$\times$3 = 8,81,526 elements takes 363 seconds to compute an index generation in MSIGT scheme. In TRSE scheme, index generation time is more in comparison with MSIGT scheme.

VII. CONCLUSION

We have proposed a MSIGT over encrypted cloud data that supports efficient index generation time. MSD radix sort algorithm is used to sort the keywords in the index file and counting sort algorithm is used to group the keywords with the same ASCII value into a bucket. The proposed MSIGT scheme is more efficient than the existing TRSE scheme

[20] and also supports a large number of data files. MSIGT scheme reduces computation and index generation overhead in comparison to earlier TRSE scheme. Finally, the proposed MSIGT algorithm is evaluated on the real data set which shows reduction in index generation time.

Future work is to reduce search time and Index storage space on encrypted cloud dataset and emphasise on extending this scheme to support different data file formats and to enhance access control for enhancing security.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] S. V. Kiran, R. Prasad, J. Thriveni, K. Venugopal, and L. Patnaik, "Cloud Enabled 3D Tablet Design for Medical Applications," in *in Proceedings of the 9th International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2014, pp. 1–6.

[3] V. Kiran, R. Prasad, J. Thriveni, K. Venugopal, and L. Patnaik, "Mobile Cloud Computing for Medical Applications," in *in Proceedings of the Annual IEEE India Conference (INDICON 2014)*. IEEE, 2014, pp. 1–6.

[4] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," http://www.techcrunch.com/2006/12/28/gmail-disasterreports-of-mass-email-deletions, 2006.

[5] T. Amazon S3, "Amazon S3 Availability Event: July 20, 2008," *http://status.aws.amazon.com/s3-20080720.html*, 2008.

[6] P. D. Shenoy, K. Srinivasa, K. Venugopal, and L. M. Patnaik, "Dynamic Association Rule Mining using Genetic Algorithms," *Intelligent Data Analysis*, vol. 9, no. 5, pp. 439–453, 2005.

[7] P. D. Shenoy, K. Srinivasa, K. Venugopal, and L. M. Patnaik, "Evolutionary Approach for Mining Association Rules on Dynamic Databases," *Advances in Knowledge Discovery and Data Mining*, pp. 325–336, 2003.

[8] D. X. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," *IEEE Symposium on Security and Privacy*, pp. 44–55, 2000.

[9] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," *in Proceedings of the Advances in Cryptology-Eurocrypt 2004*, pp. 506–522, 2004.

[10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *in Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 79–88, 2006.

[11] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," *in Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, pp. 253–262, 2010.

[12] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A. L. Varna, S. He, M. Wu, and D. W. Oard, "Confidentiality-Preserving Rank-Ordered Search," *in Proceedings of the 2007 ACM Workshop on Storage Security and Survivability*, pp. 7–12, 2007.

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

[14] 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," *6th Annual International Conference on ICT: Big Data, Cloud and Security (ICT-BDCS 2015)*, pp. 43–48, 2015.

[15] X. Sun, Y. Zhu, Z. Xia, J. Wang, and L. Chen, "Secure Keyword-based Ranked Semantic Search over Encrypted Cloud Data," *in Proceedings of the Advanced Science and Technology Letters(MulGraB 2013)*, vol. 31, pp. 271–283, 2013.

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

[17] C. Orencik, M. Kantarcioglu, and E. Savas, "A Practical and Secure Multi-Keyword Search Method over Encrypted Cloud Data," *in Proceedings of the IEEE Sixth International Conference on Cloud Computing (CLOUD)*, pp. 390–397, 2013.

[18] X. Sun, X. Wang, Z. Xia, Z. Fu, and T. Li, "Dynamic Multi-Keyword Top-k Ranked Search over Encrypted Cloud Data." *International Journal of Security & Its Applications*, vol. 8, no. 1, pp. 319–332, 2014.

[19] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. Hou, and H. Li, "Verifiable Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 3025–3035, 2014.

[20] J. Yu, S. J. T. P. Lu, Y. Zhu, G. Xue, and M. Li, "Toward Secure Multikeyword Top-k Retrieval over Encrypted Cloud Data," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 239–250, 2013.

[21] S. L. Pallickara, S. Pallickara, and M. Zupanski, "Towards Efficient Data Search and Subsetting of Large-Scale Atmospheric Datasets," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 112–118, 2012.

[22] S. Ruggieri, "Efficient c4. 5 [Classification Algorithm]," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 438–444, 2002.

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