Secure and Privacy Preserving Mail Servers using Modified Homomorphic Encryption (MHE) Scheme

A Technique for Privacy Preserving Big Data Search

Lija Mohan[1], Sudheep Elayidon M[2]
Division of Computer Science, School of Engineering,
Cochin University of Science & Technology,
CUSAT, Kochi, Kerala, India

***Abstract*—Electronic mail (Email) or the paperless mail is becoming the most acceptable, faster and cheapest way of formal and informal information sharing between users. Around 500 billion mails are sent each day and the count is expected to be increasing. Today, even the sensitive and private information are shared through emails, thus making it the primary target for attackers and hackers. Also, the companies having their own mail server, relies on cloud system for storing the mails at a lower cost and maintenance. This affected the privacy of users as the searching pattern is visible to the cloud. To rectify this, we need to have a secure architecture for storing the emails and retrieve them according to the user queries. Data as well as the queries and computations to retrieve the relevant mails should be hidden from the third party. This article proposes a modified homomorphic encryption (MHE) technique to secure the mails. Homomorphic encryption is made practical using MHE and by incorporating Map Reduce parallel programming model, the execution time is exponentially reduced. Well known techniques in information retrieval, like Vector Space model and Term Frequency – Inverse Document Frequency (TF-IDF) concepts are utilized for finding relevant mails to the query. The analysis done on the dataset proves that our method is efficient in terms of execution time and in ensuring the security of the data and the privacy of the users.**

***Keywords*—Big data; encrypted data searching; privacy preserving; homomorphic encryption; hadoop; map reduce**

I. INTRODUCTION

Today, the data is evolving at an enormous rate and Cloud Computing paved the way to economic and easy storage of Big Data. World Wide Web (WWW), Social Media, Electronic Health Records, etc. are all sources of Big Data. Since this Big Data cannot be stored and processed using single system, it is stored in multiple systems or preferably outsourced to cloud system. But, this data outsourced to a third party system like cloud raises some security challenges. NIST [23] identifies the 'Security and Privacy of the stored data' as one of the major challenge to be addressed while storing sensitive data in the cloud. According to the application requirement, methods adopted to ensure the security and privacy differs. This article explains a novel technique to implement secure Email servers that ensures the privacy of each user.

Emails are becoming the easiest, inexpensive and faster method of personal and formal communications. Many people utilize the free email service provided by Google, Yahoo, etc. Private organizations maintain their own mail servers to ensure more privacy and security of the users and data transferred. But, as the employees increase and as the size of mails increases, these organizations should maintain a good amount of infrastructure for the efficient storage which will result in a heavy maintenance cost. Cloud computing comes to the rescue here. But, ensuring the privacy and security of the users and emails is a challenging issue. "Hilary Clinton's Email Leak", "Effect of Email Leak during French Elections" [24], etc. are the result of inefficient and insecure storage and transfer of emails.

This article proposes a secure and privacy preserving technique to store, retrieve and transfer of sensitive e-mails. To ensure security, traditional encryption techniques can be utilized. Encrypt each email before passing through the network and decrypt it at the receiver side. Also, before storing the mails in cloud system encrypt it.

Thus, storage and transmission of encrypted mail is possible by utilizing existing well known cryptosystems. But, search and retrieval of specific mail is the difficult part. Since, all mails are stored in an encrypted form, the direct solution is to download all the mails to the client machine, decrypt them and find the matching mails. But, this will consume a large bandwidth and hence, not at all an economic solution, considering the pay-as-you-use pricing model of cloud. Also, if there are too much mails, download and decryption of each mail will be a time consuming task and will not be feasible, if the client machine does not have much processing capability.

The scenario given in Fig. 1 illustrates the need for secure email server. Alice is working in ABC Company which processes information dealing with the national security. They maintain their own mail server for transfer of mails between their employees. The mail server is hosted on a Cloud system. Hence, to ensure the security, mails are stored in an encrypted form. Later, to retrieve all mails related to "Mission X", either Alice need to download all mails to her system, decrypt them and search or decrypt all mails at the Cloud system and search and retrieve only the specific mails. The former method wastes a lot of bandwidth and later results in security violation as decryption is done at a cloud machine.

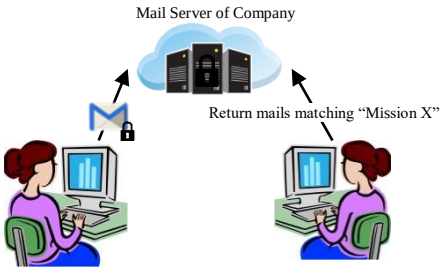


Fig. 1. Scenario illustrating the need for secure Email server.

II. Related Work

Encrypted data searching is partially made possible through different techniques like Property Preserving Encryption [15], Searchable Symmetric Encryption [16], [17], Homomorphic Encryption [10], etc. But none of these methods have been found to be efficient and practical for real word applications. Hence, based on the application context, an algorithm is selected and modified to meet the privacy and security constraints. Statistical and access pattern leakage makes PPE schemes less adoptable to Cloud [1], [2]. SSE schemes are preferred over PPE for more storing more sensitive data but at the cost of complex operations like pairing, elliptic curves, etc. Attribute Based Encryption, Identity Based Encryption, etc. helps to restrict the access to the documents but does not support content searching. Fully Homomorphic Encryption scheme put forward by Gentry in 2012 [11]-[13] is considered as a holy grail for encrypted data operations but no practical methods have been put forth which can be directly applied to any application. Oblivious RAMS [19] are another concept to prevent access leakage but with a higher implementation cost.

III. Background

The authors utilise the well-known techniques for information retrieval, like vector space model [3] and TF-IDF [4], for retrieving the relevant documents. The search similarity index thus generated is encrypted using homomorphic encryption [5] scheme and encrypted functions are applied on it to retrieve the similar document indices. This list is then sent to the client side and the ranking is done there by decrypting the obtained indices and sorting them based on their similarity score.

A. TF-IDF Calculation

Term Frequency – Inverse Document Frequency is a statistical measure used to evaluate the importance of a word in a document, or a corpus. Term Frequency implies the cardinality of occurrence of each word in a document and Inverse Document Frequency implies the importance of a word in the entire corpus.

$$TF_{i,j} = N_{ij} / \sum N_{kj} \quad (1)$$

Where TF_{ij} implies the term frequency of *an* i^{th} word in j^{th} document, N_{ij} implies the frequency of occurrence of i^{th} word in j^{th} document and $\sum N_{kj}$ implies the total number of words in the j^{th} document. Since we are dealing with Big Data, we utilise a normalised TF value for further evaluations.

$$TF_{nij} = TF_{ij} / \max(TF) \quad (2)$$

Where TF_{nij} implies the normalised TF value for the i^{th} word in the j^{th} document and max(TF) implies the maximum value for TF obtained for any word in the document collection.

$$IDF_i = 1 + log(D/|F_i|) \quad (3)$$

Where, |D| implies a total number of documents in the corpus and $|F_i|$ implies a total number of occurrence of terms in the corpus.

B. Vector Space Model

Vector Space model [6]-[9] represents text documents in rows and columns, where the rows are distinct words, and the columns are documents in the corpus and each cell represents the degree to which each word belongs to a document. TF-IDF is used as the metric to represent the degree of relevance of words in a document. This model represents documents and words as a vector.

Document collection, $D_t = (d_1, d_2, d_3, \ldots, d_t)$

Word Collection, $W_k = (w_1, w_2, w_3, \ldots, w_k)$

If D_t is arranged in columns and W_k in rows, each cell, C_{tk} represents the similarity score.

When a query comes with x words, $Q_x = (w_1, w_2 \ldots w_x)$, the similarity of the document is identified by (4).

$$Similarity\ Score,\ S_t = \sum_{i=1} C_{it} * B_i \quad (4)$$

Here, B_i has a value 0/1, depending on whether the word is present in the query list or not.

After obtaining the similarity score for 't' documents, they are ranked in order to find the most similar documents.

IV. Modified Homomorphic Encryption (MHE) Scheme

According to Gentry's Homomorphic Encryption scheme using ideal lattices, the encryption scheme is c = pq+2r+m and the decryption scheme is m = (c(mod p))mod 2. Before encrypting any message, it should be converted to binary and each bit is encrypted by using this formula. This is a generalized method to be adopted if we do not know the value of message to be encrypted. But if we know this message range prior, the complexity of this scheme could be greatly reduced. Binary conversion and bit by bit encryption can be replaced by a single step encryption and decryption vice versa. Also, the number of bits needed to store the encrypted value will be drastically reduced to approximately 1/||n|| where n is the number of bits in the binary representation of the message.

A. Modified Homomorphic Encryption (MHE) Algorithm

Let m ranges from 1 to n, then set $s = 2^{2||n||}$. The secret key, p should be a large number of the order of $O(s^3)$. The noise parameter, r will be a smaller value compared to s and p; i.e. p >>>s >>r.

Encryption: Encrypt(SK, m):

Given m ε Z_n and the secret key p, choose a random value for r and q.

Cipher Text, c = pq + sr +m
Decryption: Decrypt(SK,c):

Given the secret key, 'p' and the cipher text, 'c' then output is, **m = (c mod p) mod s.**

B. Proof of Correctness for MHE Algorithm

Proof of correctness for decryption

$$\mathbf{m} = (c \bmod p) \bmod s$$
$$= ((pq+sr+m) \bmod p) \bmod s$$
$$= m \bmod s \text{ (since } p > s.r)$$
$$= m$$

Proof of correctness for homomorphic addition

$$m_1 + m_2 = ((C_1 + C_2) \bmod p) \bmod s$$
$$= ((pq_1+sr_1+m_1 + pq_2+ sr_2+m_2) \bmod p) \bmod s$$
$$= ((p(q1+q2)+s(r1+r2)+m1+m2) \bmod p) \bmod s$$
$$= (s(r1+r2)+m1+m2) \bmod s \text{ (since } p >> s(r1+r2))$$
$$= m_1 + m_2$$

Proof of correctness for homomorphic multiplication

$$m_1 * m_2 = ((c_1 * c_2) \bmod p) \bmod s$$
$$= ((((pq_1+sr_1+m_1 * pq_2+ sr_2+m_2) \bmod p) \bmod s$$
$$= (s^2 r_1 r_2 + sr_1 m_2 + sr_2 m_1 + m_1 m_2) \bmod s$$
(since $p >> s^2$.noise)
$$= m_1.m_2$$

V. System Design

Secure Mail Servers encrypt each mail before passing it through the network. Public Key Cryptosystem powered by LDAP is utilized for this. For storage of mails, as well as for the secure transfer of mails, traditional cryptographic techniques are utilized, as it is found to be more efficient and less time complex. For encrypting the mails, AES is utilized. Each mail is encrypted by user's secret key and uploaded to cloud. To fetch a mail, the same key is used for decryption. Also, while sending a mail, it is encrypted by receivers' public key using RSA encryption system. Receiver can use his secret key to decrypt and view the contents of the mail.

Each mail will be stored in the cloud system in an encrypted form. To search and retrieve the matching mails from this encrypted domain, a vector space is generated and encrypted using the Modified Homomorphic Encryption scheme (discussed in Section 3.1). A two round search and retrieval strategy is followed. During the first round, a trapdoor is generated with the query keywords and is used to calculate the encrypted score of each mail. Cloud system will return the Mail-ID along with the encrypted score to the user. User will decrypt the scores, rank them and send the top-K Mail-IDs to the cloud. Cloud will now send the corresponding encrypted mails to the user in the second round of communication. Fig. 2 illustrates the two round search and retrieval scheme. As explained in Section 1, secure mail storage and transmission is achieved using traditional cryptosystems. How to securely retrieve the relevant mails are discussed in the next section.

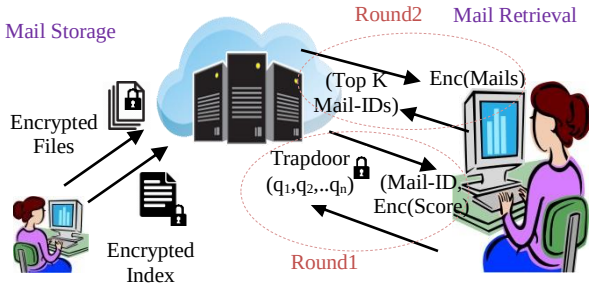


Fig. 2. Two round search and retrieval scheme.

A. Secure Mail Storage for Secure Retrieval

To implement secure ranked mail retrieval, we adopt the indexing technique used in Information Retrieval. Before encrypting a mail for the secure storage, generate the vector space model filled with TF-IDF values. The TF-IDF value is then normalized using min-max normalization to put within the rage 1 to n. Each cell value is then encrypted using the MHE scheme described. Words and Mail-IDs are removed from this index and its order is kept as a key for recovering the relevant files. The MHE encrypted index is uploaded to cloud along with the encrypted mails. Each time when a mail is sent or received, only this index has to be updated and the changed order of words and mail IDs will be made available to the owner.

B. Secure and Ranked Mail Retrieval

To search for a particular mail containing some query keywords Q = $(q_1,q_2,\ldots q_n)$, a string, S is generated which is a combination of 0s and 1s. The length of the string will be equal to the size of the Wordlist. Corresponding to each word in the Wordlist, if that word is present in the query, it will be set otherwise it will be unset. Each bit of this string is then encrypted using MHE to form the trapdoor.

for each $word_i$ in the Wordlist

$$\text{If}(word_i \text{ in } Q)\ S_i = 1; \text{ else } S_i = 0 \qquad (5)$$

On receiving the trapdoor, the cloud will do multiplication and summation on the index to obtain the encrypted scores corresponding to each column using equation 6. Due to the additive and multiplicative homomorphic property of MHE, the operations done on this encrypted data will be homomorphic to the operations done on raw data. The list of encrypted scores thus obtained is returned to the user.

$$Similarity\_Score,\ SS_m = \sum_{i=1}^{w} TF\text{-}IDF_{id} * Tw_i \ . \qquad (6)$$

User will decrypt the score with his secret key and rank them to identify the top-K matching mails. The corresponding mail IDs are sent to the cloud. The cloud will return the encrypted mails which are then decrypted at the client side. Thus a two round communication is initiated between user and cloud system to retrieve the matching mails. Decryptions take place only at the client side to ensure absolute security. Also, compute intensive operations like score calculation take place at the cloud which ensures efficiency.

Ranking of scores involve finding the greatest 'k' values from the list. To reduce the complexity of ranking procedure using sorting techniques, an efficient swapping strategy is adopted. This method is less time intensive compared to traditional sorting. Ranking is achieved using Algorithm 1. There is no need of sorting the entire list. Suppose we need to identify the top k scores, add k values to a 'topKlist' from the input values arranged in ascending order. For each next value in the input array, if it is less than the first value of 'topKlist', discard it otherwise, remove the first value of the list and add the new value to the correct position in the 'topKlist'. After each value in the input array is scanned, the 'topKlist' will contain the most relevant k mails arranged in ascending order. This algorithm reduces the complexity to O(mk) where m is the total number of mails and k is the number of mails to be retrieved. Complexity can further be reduced by applying a tree structure.

C. Improving the Ranking of Mails

Apart from the content similarity of the mail with the query keywords, there are other factors that affect the ranking of similar documents. For example, if a user has marked one email as 'important', then such mails shall be given some weightage even if their similarity score is a bit less. This is achieved by adding one more row to the vector space for including the weight of the mail. If the mail has been marked as 'important' by the user, the filed will be set 1 else 0. The value can be increased or decreased based on application requirement. The same technique can be applied to emails tagged as spam, promotions, etc.

The entire stages of the Secure Index Generation are summarised below:

1) Setup(λ): Based on the security parameter, λ the data owner generates the secret key SK.

2) IndexBuild(DocCollection,SK): Documents are arranged in vector space model after applying IR techniques like stemming and stop word elimination. The index is homomorphically encrypted to generate a secure index, I_w with height w and width m, using the secret key, SK. Then, I_w is uploaded to cloud server along with other encrypted mails.

3) TrapdoorGenerate(Query,SK): The query keywords obtained from user, Qn are arranged into a Boolean Query vector form Q_w, where $Q_j = 1$ if w_j is present in Q_n else 0. Q_w is then homomorphically encrypted using SK to form the trap door, T_w. T_w is sent to the cloud server.

4) ScoreCalculate(T_q,I_e): Encrypted score, 'es' of each mail is calculated using equation 1. Resulting vector will be $SS_m = (es_1, es_2, \ldots, es_m)$

5) Rank(SS_m,SK,n): Encrypted Scores are decrypted at client machine using secret key, SK and retrieve the actual scores, $S_m=((fid_1,s_1),(fid_2,s_2),\ldots,(fid_m,s_m))$. Sort the scores to find the top *n* similar mails matching with the query.

6) Retrieve Top Matching Files: The top-K ranked document ids are sent to the cloud server and it returns the encrypted documents to the clients, which can then be decrypted to view the mail contents.

Algorithm 1: Top-K Similar Document Select Algorithm (S_d,K)

Input :
S_d : list containing scores of each file $S_d=((fid_1,s_1),(fid_2,s_2),\ldots,(fid_d,s_d))$
K : number of files to be retrieved.
Output:
$TopList_K$ = Top K-Relevant Files

```
1.  Initialize: TopList_K = NULL
2.  For each item Ɛ S_d
3.      If length(TopList_n) < K
4.          Add item to TopList_n in ascending order of the score
5.      Else
6.          If(item['score'] > TopList_0['score'])
7.              Replace TopList_0 with item
8.              Sort first K elements in TopList_n in ascending order.
9.          Else
10.             Discard the item
11.         End IF
12.     End IF
13. End For
14. Return TopList_n
```

VI. Security Analysis

The security of the proposed scheme should guarantee that the outsourced data is safe at the third party storage. The cloud server that we consider is always an honest, but curious system. Hence, the data as well as the related information like index, keywords, etc. should be protected against statistical leakage, access pattern identification and term distribution. The overall security of our information retrieval system depends on the security of the proposed encryption scheme and the distributed implementation of the index creation and retrieval phase.

A. Security of the Homomorphic Encryption Scheme used for Securing the Index

The proposed MHE scheme is secure and can be explained based on the approximate GCD problem. Consider the approximate-GCD instance $\{x_0, x_1, \ldots x_t\}$ where $x_i = pq_i + r_i$.

Known attacks on the approximate-GCD problem for two numbers include brute forcing the reminders, continued fractions, and Howgrave-Graham's approximate-GCD algorithm [14].

A simple brute-force attack is to try to guess r_1 and r_2 and verify the guess with a GCD computation. Specifically, for r_1', r_2' ε $(2^{\eta-\rho}, 2^{\rho})$, set $x_1' = x_1 - r_1'$, $x_2' = x_2 - r_2'$, $p' = GCD(x_1', x_2')$.

If p' has η bits, output p' is a possible solution. The solution p will definitely be found by this technique, and for the parameter choices, where ρ is much smaller than η, the solution is likely to be unique. The running time of the attack is approximately $2^{2\rho}$.

Attacks for arbitrarily large values of t include lattice-based algorithms for simultaneous Diophantine approximate [20], Nguyen and Stern's orthogonal lattice [21], and extensions of Coppersmith's method to multivariate polynomials [22].

Apart from the security of the homomorphic encryption, our scheme utilised word order and document order as the keys for correct retrieval. W words and D documents can be arranged in W! X D! ways and as W or D increases, the complexity increases.

B. *Security of the Index Creation and Information Retrieval Scheme*

The proposed scheme uses homomorphic encryption to secure the index and without decrypting the index at server side, the encrypted similarity score of the documents is identified and returned to the client side. Hence, the method ensures that the data is secured against statistical and access pattern leakage. Suppose, if two queries contain the keyword q, then the word vector v_q in W will be set to 1, and will be homomorphically encrypted with two different keys K_1, K_2, which will yield two different cypher values, C_1 and C_2. Hence, seeing the values of the cypher text, we cannot predict the keywords that are searched and the frequency or order of accessing different keywords. Also, to prevent access pattern leakage, apart from k relevant files, k irrelevant files are also retrieved which reduces the chance of probabilistic approaches to find the file content.

The proposed scheme hides the term distribution, as tf-idf values are normalised, encrypted and stored in the cloud. Since, the encryption is not ordered preserving, depending on the absolute value of weights, a relevance of documents cannot be identified. Thus, the term distribution, as well as the inter distribution, is hidden from third party.

Fig. 3 illustrates the term distribution of the terms "data" and "resources" with other terms in the dataset.

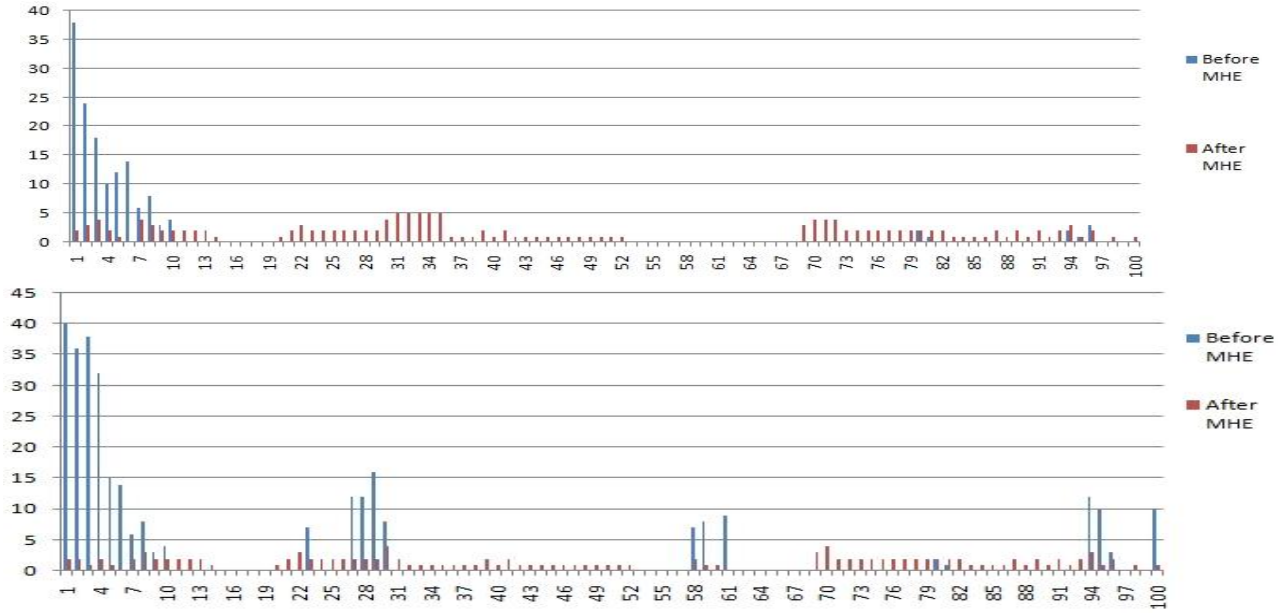


Fig. 3. Distribution of similarity relevance of 142 terms with (a) "data", and (b) "resources" before and after FHEI in the 20 Newsgroups data set.

Complex operation like encrypted score calculation is done at the cloud server and ranking of the scores is done at the client side thus ensuring the security of the data. Also, the privacy of the user is ensured by encrypting the query before sending it to cloud server.

VII. Accelerating MHE Implementation using Map Reduce

Distributed processing using Map Reduce over Hadoop accelerates the speed of execution of the index creation and retrieval stages. Fig. 2 illustrates the working of MHE using Map Reduce.

Phase 1: Encrypted Index Creation

Stage 1: Inverted Index Creation with the frequency of occurrence.

An inverted index is a data structure which stores the details of mapping from words to files. After forming the inverted index, we need to form a vector space model with each cell containing TF-IDF values. In order to simplify the vector space generation stage, we calculated the frequency of occurrence of each word in each document, simultaneously

with the inverted index creation. This list is kept separately so that it can be re-used when some modifications happen in the input document collection.

Stage 2: Encrypted Vector Index Generation

After obtaining the output from Stage 1, each Mapper for the next stage will be assigned with finding the TF-IDF of each cell. TF can be obtained from the stage1 output and DF, by summing all values. TF-IDF is calculated as per (5). After obtaining the TF-IDF value, MHE is applied as per (6) to form the encrypted value. The entire output from different mappers is then merged to form the final encrypted vector space index. This is then uploaded to cloud. Stage 2 does not require a Reduction stage. Fig. 4 illustrates the details of Map Reduce Stages. Encryption of each document can also be done efficiently by adding one more Mapper stage.

Phase 2: Ranking to retrieve K-Similar Documents.

To retrieve top k similar documents, in an ideal case, assign the task to K mappers with D/K input, where D is the total number of documents. Each mapper evaluates Algorithm 1 to find the most similar document. Collecting the output from K Mappers will give the top K matching documents. There is no Reducer needed in this case. If only N (N<K) mappers are available, assign D/N input to each mapper and evaluate Algorithm 1 to get matching K/N documents. The reducer will select top K from the final result.

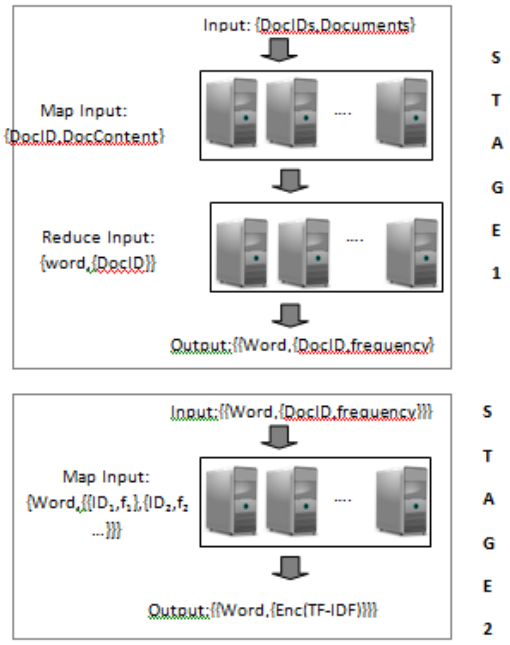


Fig. 4. Map reduce implementation of encrypted vector index creation.

VIII. Experimental Setup and Evaluations

The experiment is evaluated on 10 node Hadoop cluster setup on Amazon Web Service (AWS). Namenode is a t2.large instance. Secondary namenode and datanodes are t2.micro instances. All machines are Ubuntu 14.2 installed with OpenJDK 1.7 and Hadoop stable version 1.0.2. HDFS Replication factor is set to 3 and HDFS block size is 8MB.

A. Dataset

The dataset used for testing is Thomson Reuters Text Research Collection (TRC2). The dataset contains 1,800,370 stories which occurred at period 01-01-2008 00:00:03 to 28-02-2009 23:54:14. The size of the dataset is 2,871,075,221 bytes. TRC2 is a single long file with date, headlines and stories stored in comma separated form. To match our testing requirement, we split this large file into multiple files where each file is named with a date in ddmmyyy.txt format and the content of that file is the headlines and stories on that particular day. Thus 419 files have been generated where each file size ranges from 8MB to 16MB. To store and retrieve these small files efficiently from Hadoop Distributed File System, BAMS [18] technique is followed.

B. Performance Analysis

The entire scheme is broadly divided into 2 phases; the Documents Upload phase and the Document retrieval phase. Performance of each phase is separately analyzed and explained.

a) Performance of the Initialization phase

Initially, we need to run the setup (λ) algorithm to derive the public and secret keys needed for encryption and decryption. To reduce the tradeoff between security and efficiency, we fixed the value of λ as 128. Secret key will be a value between $[2^{\eta-1},2^{\eta}]$. Thus the complexity of this stage will be $O(\lambda^{\eta})$ which is a constant, as λ is constant.

The index building stage involves tf-idf calculation and homomorphic encryption. To reduce the execution time of index building of large data, we implemented a distributed map reduce parallel programming model that reduces the complexity to linear. Also, tokenization, stemming and stop word elimination is done to reduce the volume of keywords to be indexed. To update the documents, re-iteration of the entire index building stage is needed and to avoid such a scenario, we store idf values separately and hence only the updated part of the file needs to be re-evaluated to find tf-idf of the corresponding words. Encryption can be implemented in O(dw) time, where d is the number of documents and w is the number of words. Index generation and retrieval stage is accelerated by following a MapReduce distributed implementation. Time needed to generate the index is same for all methods whereas time needed to encrypt the index using traditional SSE, proposed MHE and MHE implemented using Map Reduce is illustrated in Fig. 5.

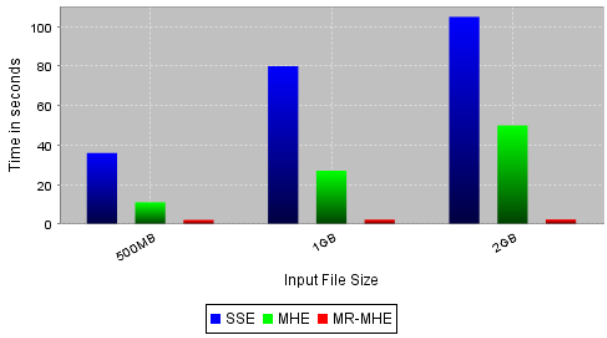
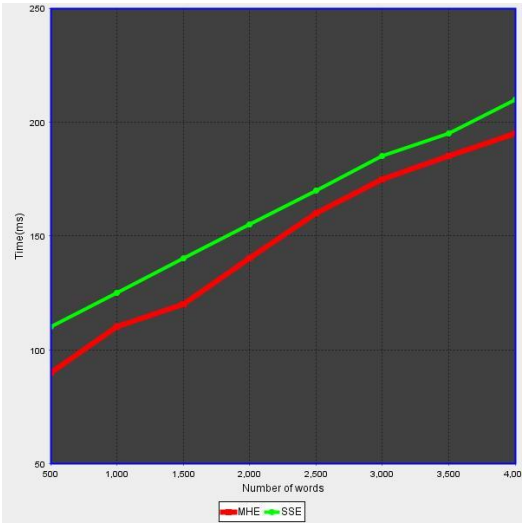


Fig. 5. Comparison of execution time for encrypted index generation.

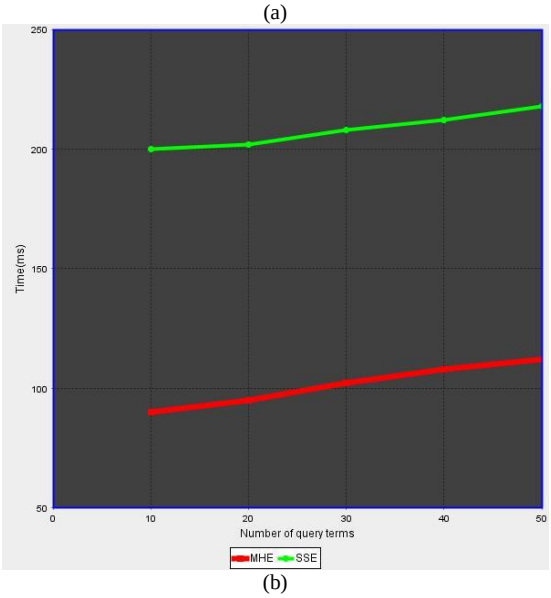
b) Performance of the Similar Document Retrieval Phase

The retrieval phase includes different stages like Trapdoor generation, Score Calculation and Sorting & shuffling to identify the top-k documents. The complexity of our proposed scheme is highly dependent on the retrieval phase, as this has to be repeated each time a user posts a query. Hence, we parallelize the most time-consuming retrieval phase i.e., sorting and shuffling of top-k results.

Trapdoor Generation involves the binary conversion of a posted query and the homomorphic encryption of each bit. If the query contains n keywords, then the complexity will be O(n). Fig. 6 illustrates the time needed to compute the Trapdoor by employing homomorphic encryption as well as traditional searchable symmetric encryption (SSE), by varying the number of total distinct words in the document collection and the number of terms in a query. It is well observed that, the execution time is approximately half for our proposed modified homomorphic encryption scheme (MHE).



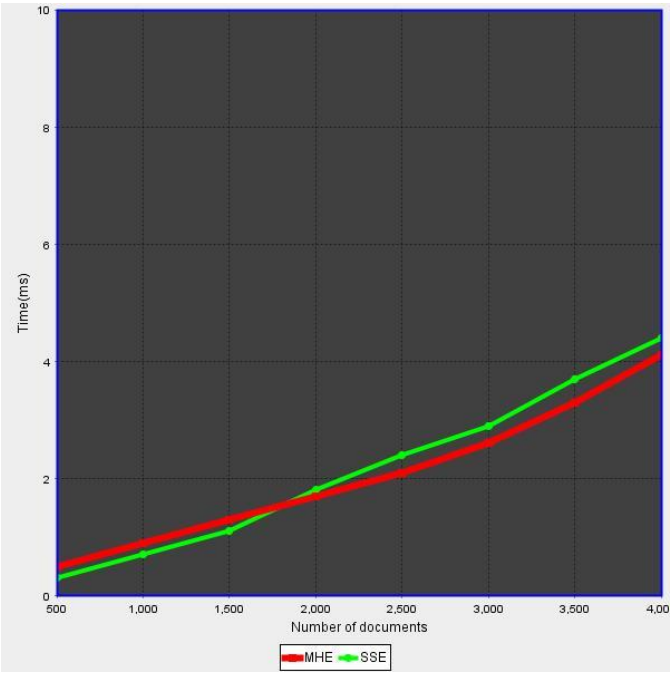
(a)



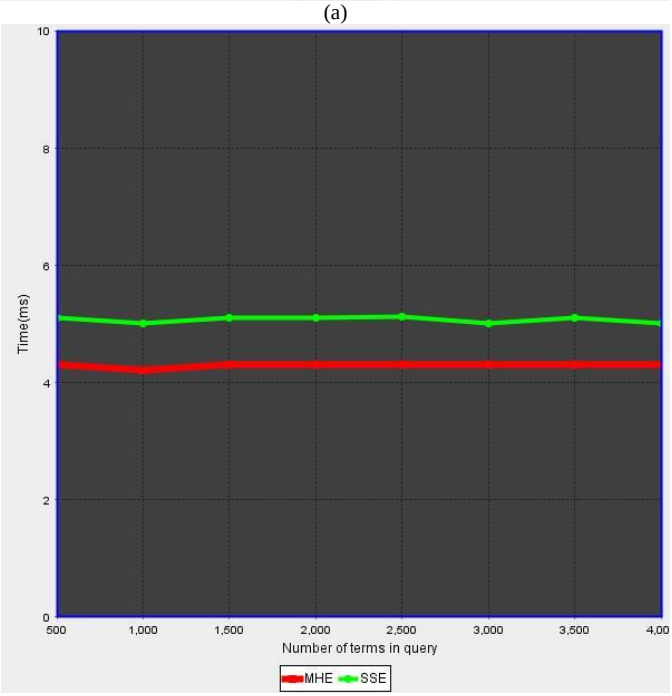
(b)

Fig. 6. (a) Execution Time by varying the total number of words in the document collection; (b) Execution Time by varying the total number of terms in the query.

To calculate the encrypted similarity score, the inner product has to be performed. This calls for w multiplications and d additions, where w is the number of words and d is the number of documents which leads to a complexity of O(wd). Here, the execution time varies with a variation in the number of query terms and documents. The comparison is illustrated in Fig. 7. It is well observed that, for the MHE scheme, the execution time is increasing almost linearly, whereas for SSE, it is an exponential increase.



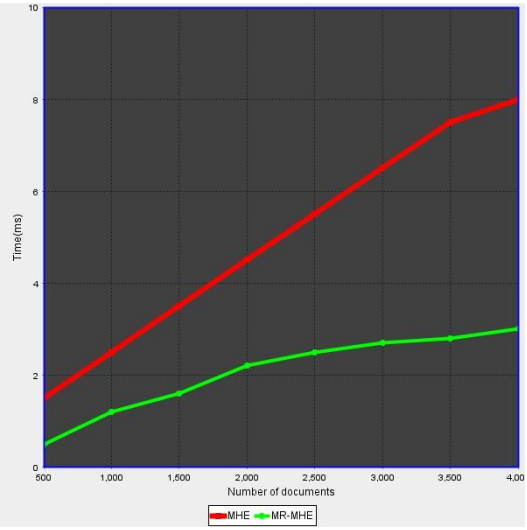
(a)



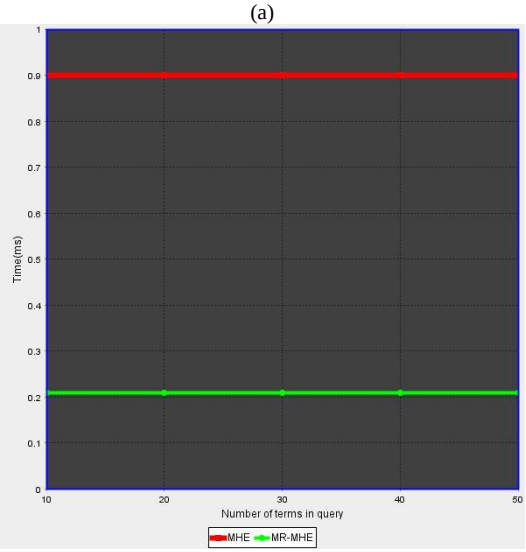
(b)

Fig. 7. (a) Execution Time by varying the total number of words in the document collection; (b) Execution Time by varying the total number of terms in the query.

Decryption of scores to obtain the similarity score is done at client side and the number of terms to be decrypted depends on total number of documents in the collection. Hence, the complexity will be utmost O(d). If there are too many documents, then distributed parallel processing can be employed to decrypt the terms. Fig. 8 illustrates how the decryption time of normal MHE scheme and MHE scheme using Map Reduce Programming Model (MR-MHE) varies with the number of documents and the number of terms in the query. Map Reduce implementation always transforms the execution time to a linear scale. Here, we implemented a cluster with only 10 nodes. The number of nodes is inversely proportional to the execution time. Hence, to decrease the execution time, the nodes can be increased. But, if the collection contains only less number of documents, map reduce processing will result in an overhead.



(a)

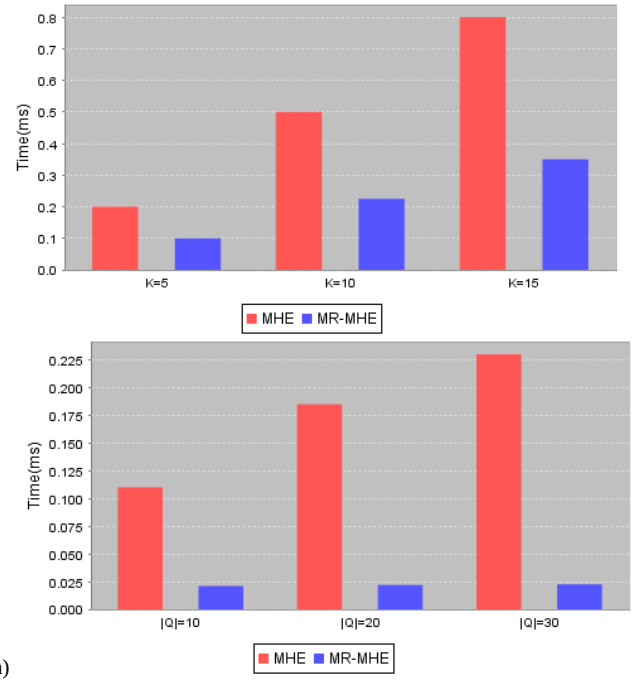


(b)

Fig. 8. (a) Execution Time by varying the total number of words in the document collection; (b) Execution Time by varying the total number of terms in the query.

Ranking and shuffling of File identifiers based on similarity score is the last stage to be executed, to identify the most similar documents. Modifying the sorting algorithm as described in Algorithm 1 itself will reduce the execution time to O(d.n) and by introducing MR programming, it can be again reduced to O(d). More reduction is possible by introducing heap tree implementation. Comparison of the execution time is shown in Fig. 8.

Fig. 9(a) shows how distributed processing improves the execution time of ranking, with a variation in k, where k denotes the number of similar files to be retrieved by the user. Here, the number of documents is set to 1000. Then, Fig. 9(b) illustrates how the performance of MR-MHE improves, with an increase in the query terms. As the number of query terms change, there is no much observable difference in execution time, as the query is distributed and evaluated. Hence, the algorithm is more scalable when the Map Reduce programming model is adopted for the implementation of our proposed scheme.



(a) (b)

Fig. 9. (a) Execution time by varying K; (b) Execution time by varying the number of query terms.

Scalability of the proposed MHE scheme is evaluated using SpeedUp metric. The SpeedUp factor defines the ratio of time needed to execute an algorithm in one machine, to the time needed to execute it on *N* machines. In an ideal case, the method is considered scalable, if the speedup factor remains constant for different values of *N*.

SpeedUp, $S_u = T_1/T_N$ ..... *(7)*

Fig. 10(a) illustrates the change in execution time for varying values of data set size. The speedup for the same is illustrated in Fig. 10(b). From the figure, it is clear that, even if the data size increases, there is no much variation in the execution time, as the number of nodes increase. Thus the

algorithm is becoming more scalable, and approaching ideal values with the increase in data size.

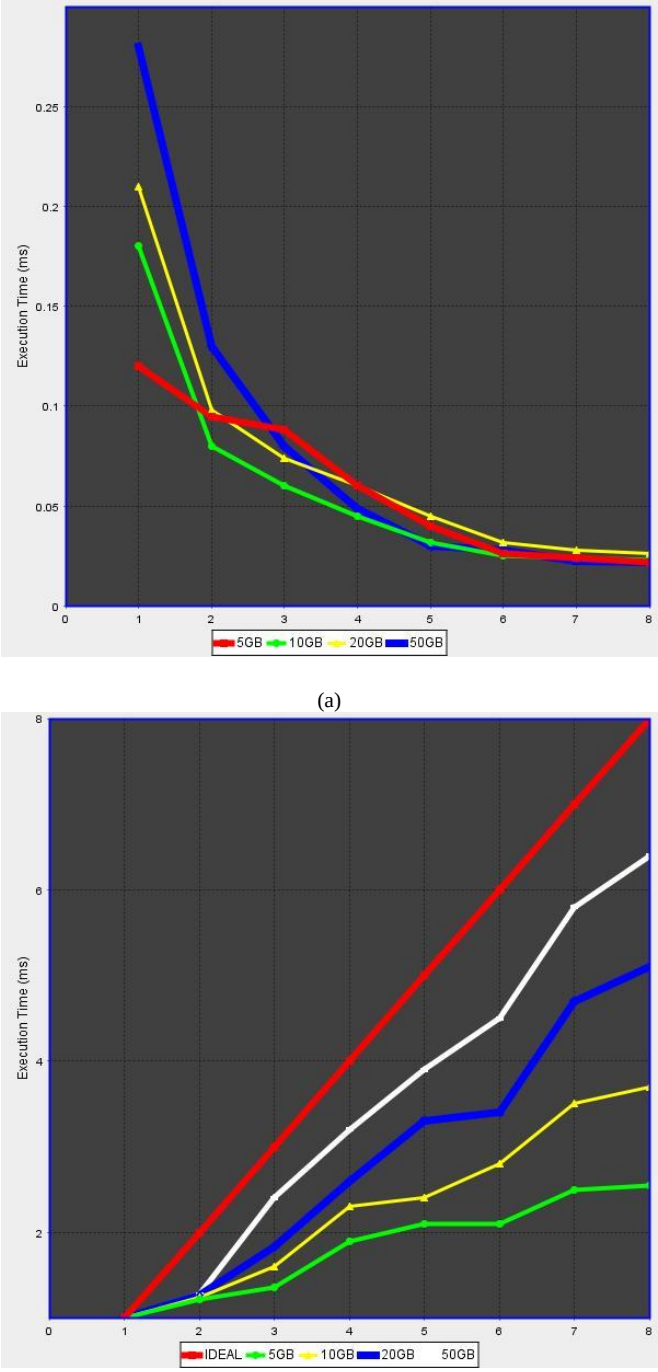


Fig. 10. (a) Execution time by varying data size; (b) Speedup.

C. Communication Overhead

The core of our approach is the homomorphic encryption of vectored index which eliminates the need of transferring the entire index to the client side for decryption and ranking. The score is calculated at the server side itself and only the encrypted scores are forwarded to the client for ranking. Consider there are 100 files and 1000 distinct keywords. Then the size of index file to be transferred for traditional SSE will be approximately 100x1000x1024 bits equals 12MB, if each cell value is set to 1024 bits. But, for the Modified Homomorphic Encryption Scheme, it will only be 100x1024 bits which are equal to .01MB. Hence, there is a large variation in the amount of data to be transferred through the network, when we compare SSE and MHE.

IX. Conclusion and Future Works

The authors have proposed a novel scheme for the implementation of encrypted mail storage and retrieval based on similarity relevance. A modified and practical version of Homomorphic encryption scheme has been utilised and the execution is accelerated, by introducing distributed Map Reduce programming model. The scheme supports multiple keyword queries, ranking of mails based on user ranking (‘important’,’spam’ etc.) and text matching by utilising most of the basic techniques in information retrieval, like vector space model, TF-IDF etc. Analysis done on the MHE scheme proves the correctness and security of the proposed scheme. The entire scheme is evaluated on a live Hadoop cluster, and proven to be efficient, secure, scalable and accurate and hence found suitable for securing a large amount of data. Currently, the updates on uploaded mailsneed revision for the entire Index creation and encryption stage, except for the TF and IDF calculation. The revised word order and file order are encrypted using the public key for each user, and passed to them whenever an update occurs. This limitation can overcome by experimenting other dynamic indexing techniques which help in storing real time data as well.

Acknowledgment

Authors sincerely thank Department of Science & Technology (DST), India for the financial support offered through INSPIRE Research Fellowship under the grant number IF140608 and Amazon Web Service (AWS) in Education Research Grant (No. 651699140108) for utilising AWS resources for free of cost.

References

[1] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, “Secure Ranked Keyword Search over Encrypted Cloud Data,” Proc. IEEE 30th Int’l Conf. Distributed Computing Systems (ICDCS), 2010.

[2] Cong Zuo, James Macindoe, Siyin Yang, Ron Steinfeld, Joseph K. Liu, "Trusted Boolean Search on Cloud Using Searchable Symmetric Encryption View Document", IEEE Trustcom/BigDataSE/ISPA, UK, 2016.

[3] Yun Zhang, David Lo, Xin Xia, Tien-Duy B. Le, Giuseppe Scanniello, Jianling Sun, "Inferring Links between Concerns and Methods with Multi-abstraction Vector Space Model" ,IEEE International Conference on Software Maintenance and Evolution (ICSME), Pages: 110 - 121, DOI: 10.1109/ICSME.2016.51, 2016.

[4] Nasser Alsaedi, Pete Burnap, Omer Rana, "Temporal TF-IDF: A High Performance Approach for Event Summarization in Twitter", IEEE/WIC/ACM International Conference on Web Intelligence (WI), Pages: 515 - 521, DOI: 10.1109/WI.2016.0087, 2016.

[5] Yasmina Bensitel, Rahal Romadi, "Secure data storage in the cloud with homomorphic encryption", 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), Pages: 1 - 6, DOI: 10.1109/CloudTech.2016.7847680, 2016.

[6] G. Salton, A. Wong, C. S. Yang, “A Vector Space Model for Automatic Indexing”, Communications of the ACM, Volume 18 Issue 11, Pages 613-620, November 1975.

[7] Wong S. K. M., Ziarko Wojciech, Wong Patrick C. N., “Generalized vector spaces model in information retrieval”, SIGIR '85 Proceedings of

the 8th annual international ACM SIGIR conference on Research and development in information retrieval, Pages 18-25,Montreal, Quebec, Canada — June 05 - 07, 1985.

[8] Waitelonis Jörg; Exeler, Claudia Sack, Harald, "Linked Data enabled Generalized Vector Space Model to improve document retrieval", The 14th International Semantic Web Conference,Pennsylvania, October, 2015.

[9] George Tsatsaronis and Vicky Panagiotopoulou, "A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness", Proceedings of the EACL 2009 Student Research Workshop, pages 70–78, Athens, Greece, April 2009.

[10] Xun Yi, Russell Paulet, Elisa Bertino, "Homomorphic Encryption and Applications", Chapter 2, Springer Briefs in Computer Science, pages 27-46, Springer 2014.

[11] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proceedings of the 41st Annual ACM Symposium on Theory of computing (STOC), pp. 169-178, 2009.

[12] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," Proc. 29th Ann. International Conference on Theory and Applications of Cryptographic Techniques, H. Gilbert, pp. 24-43, 2010.

[13] Nathanael David Black , "Homomorphic Encryption and the Approximate GCD Problem", A Dissertation Presented to the Graduate School of Clemson University, August 2014. http://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=2281&context=all_dissertations

[14] N. Howgrave Graham, "Approximate Integer Common Divisors", Proceedings of the International Conference on Cryptography and Lattices,(CaLC' 01), pp. 51-66, 2001.

[15] R. Curtmola, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," Proceedings of ACM Conference on Computer and Communications Security (CCS), pp. 79–88, 2006.

[16] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In Antoine Joux, editor, EUROCRYPT, volume 5479 of Lecture Notes in Computer Science, pages 224–241. Springer, 2009.

[17] Sanjit Chatterjee and M. Prem Laxman Das, "Property Preserving Symmetric Encryption Revisited", International Conference on the Theory and Application of Cryptology and Information Security,ASIACRYPT, Lecture Notes in Computer Science, vol 9453. Springer, Berlin, Heidelberg, 2014.

[18] Lija Mohan, Sudheep Elayidom M,"Balanced MultiFileInput Split (BaMS) Technique to solve Small File Problem in Hadoop", IEEE 11th International Conference on Industrial and Information Systems (ICIIS), IIT Roorkee, India, 2016.

[19] Lija Mohan, Sudheep Elayidom M. (2017) Encrypted Data Searching Techniques and Approaches for Cloud Computing: A Survey. In: Mandal J., Satapathy S., Sanyal M., Bhateja V. (eds) Proceedings of the First International Conference on Intelligent Computing and Communication. Advances in Intelligent Systems and Computing, vol 458. Springer, Singapore, 2016.

[20] J.C. Lagarias, "The computational complexity of simultaneous diophantine approximation problems", SIAM Journal on Computing (SICOMP), Volume 14(1),pp 196–209,1985.

[21] P.Q. Nguyen, J. Stern," Adapting density attacks to low-weight knapsacks", Proceedings of Advances in Cryptology, ASIACRYPT'05, pp. 41–58, 2005.

[22] D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities", Journal of Cryptology, volume 10(4), pp 233–260, 1997.

[23] "The NSA Scandal's Impact on the Future of Cloud Security - Data Security and Protection in the Wake of the Spying Scandal", White Paper issued by Porticor, February, 2017. http://mobility-sp.com/images/gallery/PORTICOR-The-NSA-Scandal%27s-Impact-on-the-Future-of-Cloud-Security.pdf

[24] Guidelines on Security and Privacy in Public Cloud Computing, NIST Special Publication 800-144, December 2011.