Journal of Autonomous Intelligence (2024) Volume 7 Issue 4
doi: 10.32629/jai.v7i4.1085

ORIGINAL RESEARCH ARTICLE

Aiding secure data retrieval incorporated with parallelization technique in cloud

Bharati P. Vasgi[1,*], S. M. Jaybhaye[2], Girija G. Chiddarwar[3]

[1] *Marathwada Mitra Mandal's COE, Pune 411052, India*

[2] *Vishwakarma Institute of Technology, Pune 411037, India*

[3] *Marathwada Mitra Mandal's COE, Pune 411052, India*

*** Corresponding author:** Bharati P. Vasgi, bharativasgi@gmail.com

ABSTRACT

Cloud data owners prefer to outsource data because of ease in maintenance. Data confidentiality of this outsourced sensitive data is a major task. The searchable encryption technique helps to carry out searches on encrypted data without decrypting it. In the data outsourcing environment, volume of data is increasing rapidly. Hence the time required to build the index and to carry out searches is also increasing exponentially. This makes it more difficult to build a system which is efficient, reliable and can cope up with growing data. In this paper, a parallelization technique to build the index on outsourced data is proposed. This technique minimizes the time required to construct the index. It also supports secure ranked retrieval using bucketization technique. The buckets are formed using Hadoop map reduce framework which achieves significant efficiency. The proposed method prune the keyword dataset, which helps in significant reduction in the size of index. Through extensive experiments using standard dataset, the performance of the system is validated. The experimental results show that the proposed system requires less time for index construction and hence improves retrieval efficiency.

Keywords: data outsourcing; searchable encryption; map-reduce; secure search; buketization; multi-owner; distributed index

ARTICLE INFO

Received: 1 August 2023
Accepted: 29 August 2023
Available online: 24 January 2024

1. Introduction

Recently, there is a huge requirement of data outsourcing for the purpose of cost saving. For small and medium sized organizations, the maintenance of storage and computing infrastructure is becoming difficult day by day. Also for large organizations it is becoming challenging task to find out people with high technical competency for challenging assignments. They have to invest huge amounts on this technical expertise. Therefore, data outsourcing is becoming reasonable solution for enterprises to relieve their burden. The security of this outsourced data is the major concern. Any leakage of this data results in loss of very sensitive information Yang et.al. and Jung et al.[1,2]. To maintain the security of this outsourced data, data and queries must be encrypted before they are submitted to cloud which is managed by third party.

The encryption of this definitely ensures the confidentiality but limits the simple operations on this encrypted data. Some of the very common operations like multi-keyword, range search is not supported.

All techniques discussed till now are having high computational

complexity as size of dataset grows. Hence the proposed system is very effective as it can handle large corpus by using parallelization techniques. The queries are handled efficiently by using posting lists. The buckets which are representing keywords in the dataset are pruned, so that more importance is given to significant keywords and thus users requests are handled in minimum time.

The paper is organized as follows. The related work is described in Section 2. Section 3 discusses the framework of the system. Section 4 gives design goals. Section 5 gives threat model and section 6 describes mathematical model, Section 7, 8 represents the proposed system. Section 9 describes security analysis of the system. Section 10 represents performance evaluation using standard dataset and Section 11 concludes with conclusion and future work.

2. Related work

Searchable encryption is the most frequently used technique to carry out searches on encrypted data. Many systems are discussed which uses searchable encryption are presented in Orencik et al.[3], Strizhov and Ray[4], Cao et al.[5], Cash et al.[6], Chen et al.[7] and Van et al.[8].

Curtmola et al.[9] discusses strong security definitions of searchable encryption. For keyword based searching bilinear pairings are put forth by various authors. The major drawback of this system is their computational cost. Inverted index technique is given which is used to carry out searches on sensitive outsourced data. The scenario where more interactions are expected by the user, this system is not of much use Orencik and Savaş[10] author proposes multi-keyword search that basically maps sensitive information to constant length array but supports limited ranking. The detail survey of issues in data outsourcing is discussed by Vasgi and Kulkarni[11]. Hash based mapping structures are used to retrieve sensitive information from the encrypted collection suggested by Vasgi and Kulkarni[12]. This system is very efficient as by using hash structure directly the location in the index is accessed. As currently discussed methods concentrate only on few features the proposed methods addresses single keyword, multi keyword and secure k-NN retrieval. It also tries to address Multiowner environment and distributed indexing approach.

The basic problem that secure k-NN queries handle is to retrieve the top similar k documents from the encrypted dataset without actually revealing the original content of document and query Wong et al.[13] formalizes the formal requirements for secure k-NN search and search top k relevant document to a query document without leakage of the content of query and document. His method is SCONEDB (Secure Computation ON Encrypted Data Base). In his method he uses symmetric scalar product preserving encryption (ASPE) scheme which query and data are encrypted differently. The scalar product between query and data point is calculated using ciphertext.

Range search is discussed in Shi et al.[14]. The multidimensional range query is suggested by Hore et al.[15] which utilizes the bucketization technique. In this, data is transformed into various buckets and also query is transformed into subset of buckets. One of the major drawbacks of this system is false positive rate.

3. The framework

In proposed framework of secure data retrieval using parallelization technique (SDRPT), we provide privacy preserving search in three different search models: multi-keyword search, k-NN search for documents (i.e., document similarity) and single keyword search. We consider a data outsourcing scenario that consists of three entities: data owner, two non-colluding semi honest servers and users. The big picture for the interactions between the entities is illustrated in **Figure 1**. The communication among various entities is listed.

Message Communication

1) Sharing of secret key k_s
2) Uploading secure bucket index on search server by owner 1

3) Uploading encrypted document collection on file server by owner 1
4) Uploading encrypted document collection on file server by owner *n*
5) Uploading secure bucket index on search server by owner *n*
6) Merging of multiple index on cloud server
7) Query submitted by end user
8) Top *k* relevant documents identifiers
9) Top *k* encrypted documents
10) Top *k* plaintext documents

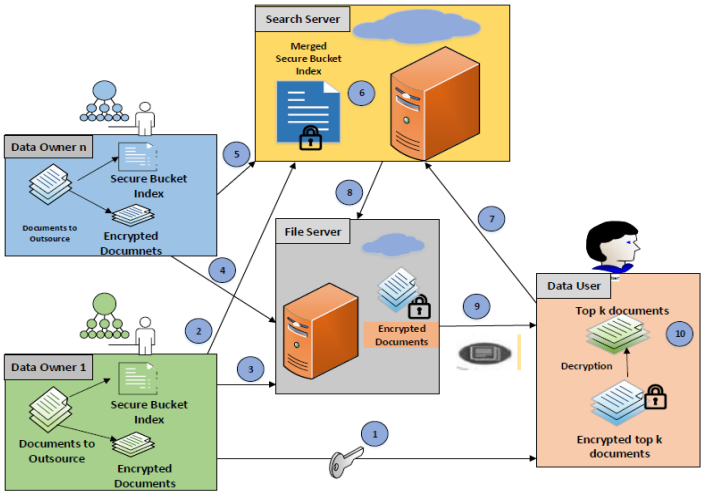

Figure 1. Architecture of proposed system.

In the proposed system, secure information retrieval is proposed which facilitates single keyword, multi keyword and secure k-NN search using map reduce architecture. The architecture shows three important entities: data user, data owner and two servers. These two servers are semi honest and non-colluding.

4. Design goals

To support secure ranked retrieval in outsourced environment, where size of corpus is growing exponentially, the proposed system is having following goals

Efficient index construction: As the size of outsourced data increases, size of indexing structures also increases. Hence conventional linear techniques for index construction are not suitable in cloud environment. The proposed method constructs the index in parallel by using Hadoop-map reduce architecture.

Support of wide range of query: The proposed system supports single keyword, multi-keyword and secure k-NN queries. The wide range of queries increases system usability.

Search efficiency: The feature set of outsourced data is pruned. This results in significant reduction in size of index and consequently increases search efficiency. The searching is analyzed with multiple cases.

Distributed indexing: The proposed system supports distributed indexing. Hence user's query is searched by distributing it to various nodes. The final results from respective nodes are merged and given to the user.

The proposed system uses stemming algorithm Baeza-Yates et al.[16], order preserving symmetric encryption algorithm Boldyreva et al.[17], secure hash algorithm Stallings.[18], DES Stallings at al.[18], TF-IDF ranking scheme.

5. Threat model

Threat models can be categorized depending upon what information is available to cloud server as:

Known ciphertext model: In this model the data available to cloud server is in encrypted form only. In proposed system, the data owner outsources encrypted document collection C_e and encrypted index I_e to cloud server.

Known background model: This model is stronger than known ciphertext model. The cloud server has additional information as compared with known ciphertext model Katal at al.[19]. This information may be the partial knowledge of scores stored in the index, access pattern, relation among various search queries and so on. With the help of this, cloud server can recognize significant information Boneh and Waters[20] and Manning and Raghavan[21].

6. Mathematical model

A mathematical model helps in describing the system by using mathematical concepts. This section explains mathematical model of proposed SDRPT technique.

- Variable
  - S_{ij}: Score of i^{th} term in j^{th} document;
  - x: normal data value;
  - p_1: Numeric plaintext value;
  - p_2: Numeric plaintext value;
  - k_s: Secret key.
- Parameters
  - C: Set of files to be outsourced, $C= \{f1, f2, \ldots, fn\}$.
  - T: Set of distinct terms derived from the file collection, $T= \{t_1, t_2 \ldots t_{m1}\}$.
  - q: Query given by user.
  - Tq: Set of terms present in query, $Tq = \{q_1, q_2, \ldots, q_{m2}\}$.
  - m_2: Total terms present in query q.
  - α: Threshold value.
  - β: Total number of buckets.
  - d_q: Query document.
  - I: Bucket Index.
  - I_e: Secure bucket index.
  - K_s: Secret key used for HMAC.
  - n: Total number of documents in the collection.
  - *tterm*: Pruned set of total features.
  - $\pi(B_i)$: Signature of Bucket B_i.
  - t_i: ith term.
  - *toutput*: Temporary output vector with structure *(docid, score).*
  - *foutput*: Final output vector with structure *(docid, score).*
- Decision variable

  α: The value of α decides the size of pruned collection. If the value is larger, resultant collection's size becomes smaller and vice-versa. The selection of α is important step as it effects on performance of the system.
- Independent function
  - S_{ij}: Score calculation of i-th feature in jth document.
  - Dec(x): Decryption of element x.

- Enc(x): Encryption of element x.
- OPE (Enc, Dec, k_s) order preserving symmetric encryption for every plaintext if ($p_1 \leq p_2$) then Enc (p_1, k_s) $\leq$ Enc (p_2, k_s).
- Mapper (documnets): (keyword, document-id, score).
- Reducer (keyword, document-id, score) (keyword, list).

- Model

$$\alpha = \{\text{High}, \text{low}, \text{Average}\} \quad (1)$$

High, low, average value indicates size of pruned dataset.

$$B_k = \left(id(f_j), (S_{jk})\right) \quad (2)$$

$$\pi(B_k) = HMAC_{ks}(SHA(B_k)) \quad (3)$$

$$VB_k = (DES\left(\left(id(f_i)\right), \left(OPSE(S_{jk})\right)\right) \Big| \; id(f_j) \in C) \quad (4)$$

7. Proposed system

This section gives details of the proposed framework. The secure search algorithm is basically consisting of secure index generation by using map reduce framework. This secure index is outsourced to search server and encrypted dataset is outsourced to file server. The encrypted index and data, prevents search server and file server to learn any secret information.

7.1. Secure index construction using parallelization technique

Secure and searchable index is created by using buckets. First of all, features are extracted from document collection. In proposed technique documents are partitioned into buckets. As secure hash algorithm is used to form bucket signature. The documents having common features share the same bucket. This helps in retrieving exact matching documents.

Every document in bucket is represented as (Document_id, score), score is calculated using following formula.

$$S_{ij} = \frac{1}{|f(w_j)|} (1 + l_n F_{ij}) \quad (5)$$

Every bucket is identified by signature of respective keyword which is calculated by using secure hash algorithm. A sample bucket creation is shown in **Figure 2**.

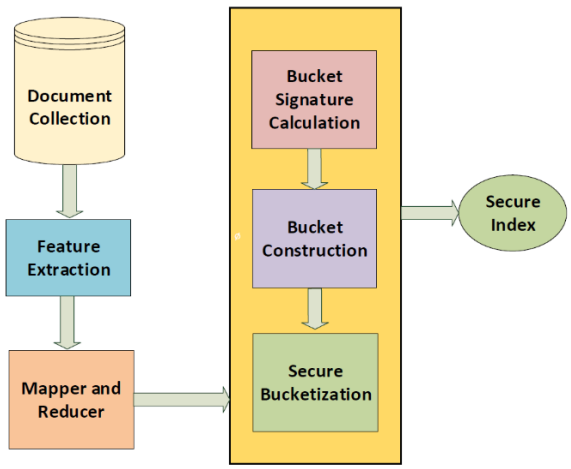

Figure 2. Secure index generation.

Since document collections are larger in size, map reduce technique is proposed for parallel index construction. The index generation system consists of three modules bucket index creation using map reduce, bucket signature calculation and secure bucket index creation.

7.1.1. Bucket construction using parallelization technique

Collections are generally very large in outsourced environment, so it is difficult to construct the index linearly. Parallelization techniques can be used for efficient index construction. The proposed system discusses index construction by using map reduce technique. map reduce is designed for large datasets. the index construction process described here is the application of map reduce. The input dataset is divided into small chunks and these chunks are processed in parallel. A master node is responsible for assigning and reassigning tasks to various worker nodes. A map and reduce phase of MapReduce divides the input task into smaller tasks so that the entire task can be executed efficiently. The details of this working are shown in **Figure 3**.

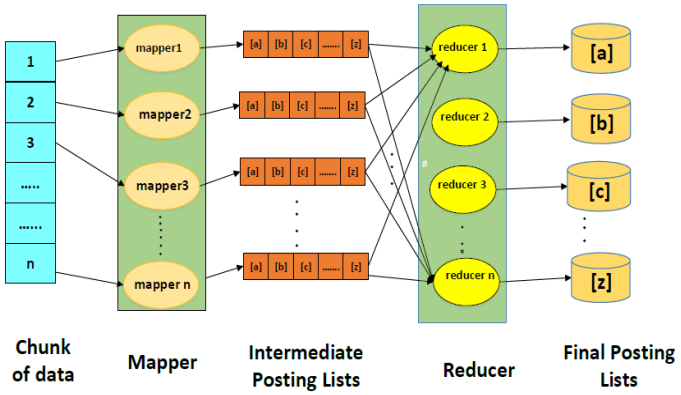

Figure 3. Bucket construction using map-reduce.

Initially the collection of data is split into small chunks such that the total work distribution is done evenly among worker nodes. These splits are assigned by master node runtime. If any worker node finishes the task master node assigns it the next chunk. If any worker node fails during the processing, its task is reassigned by master node to other available worker node.

The map phase of MapReduce maps the splits of input data into *(key, value)* pairs. In proposed system this *(key, value)* pair is *(keyword, document-id, score*). The mapper class is also known as parser. Each working parser writes its output to intermediate files called as segment file or posting file as shown in **Figure 3**. The parser designed in this process partitions the keys into *j* term partition, so that parsers can write *(key, value)* pair to respective partition. In the proposed system partitions are created in lexicographical order as shown in **Figure 3**.

In the reducer phase same keywords from all intermediate files are reduced to single list. This will make processing of the keywords fast and easy. The reducer collects all document ids from intermediate files for a particular keyword and prepare buckets for each keyword. This bucket is also called as posting file. This process is illustrated by giving a simple example as below.

Scheme of map and reduce functions

map: input → list(key, value)

reduce: (key, list(value)) → output

Instantiation of the schema for index construction

Map: document collection → list (keyword,docID,Score)

Reduce: ((termId$_1$, list(dociD, score)), (termid$_2$, list(docid, score)),……) → (bucket$_1$, bucket$_2$)

Sample example for bucket index creation

map: (d_2: internet, protocol), (d_1: WSN, internet, NIC, internet)→((internet, d_2), (protocol, d_2), (WSN, d_1), (internet, d_1), (NIC, d_1))

reduce:

((internet, (d_2, d_1, d_1)), (protocol, (d_2)), (WSN, d_1), (NIC, d_1))→ ((internet, (d_2:1, d_1:2)), (protocol, (d_2:1)), (WSN, (d_1:1)), (NIC, (d_1:1)))

Example 1: Shows the general schema of map reduce functions. Here inputs and outputs are key-value pairs and buckets respectively. All these MapReduce jobs run in parallel. As it splits index construction in smaller tasks, it can scale up to large data collection provided with the support of underlying hardware architecture.

The map function results in key-value pairs. All values i.e. document ids for a given keyword are collected into a bucket in reduce phase. This bucket is then used for secure retrieval. These two functions help in construction of bucket index. For simplicity keyword frequency is shown in the list, but real time this frequency is replaced by scores.

7.1.2. Bucket signature calculation

For all $T = (t_1, t_2, \ldots\ldots t_{m1})$, m_1 feature's signatures are calculated by applying secure hash algorithm. Thus constant length signature is generated for each bucket. Each feature is hashed with a standard secure hash function. In the feature set T, there are many features with very low relevancy scores. Use of all these features may have adverse effect on index construction and index utilization. Hence feature set is pruned by using standard threshold value α. All the features having score above α are only considered for index construction. Assume that after pruning we get β buckets. With this pruning, search efficiency will improve but may have drawback that documents with rare features cannot be searched.

After the bucket signature calculation, document identifiers are distributed to β buckets according to their signatures. Let B_k be the bucket identifier for the kth feature. The content of vector bucket B_k is given as in Equation (4). Note that each document can be mapped to various buckets depending upon the dominant features they hold.

7.1.3. Secure bucket index creation

Bucket identifiers and bucket data holds very sensitive information. This information must be encrypted before outsourcing. The bucket identifiers contain very useful information like the features it holds and the features it does not hold. To avoid adversary to learn from this, bucket signatures and contents are encrypted using cryptographic techniques. Bucket content e.g., document identifiers are encrypted using standard data encryption algorithm and scores are encrypted using Order Preserving Symmetric Encryption. Also bucket signatures are secured using HMAC. The key k_s is securely shared between data owners and data users. Since the key k_s is not known to search server and file server, the system is secure against brute force attack. The secure bucket identifier is denoted as in Equation (3). The encrypted content of the bucket vector is denoted by Equation (4).

Since scores are very sensitive information as they furnish importance of keyword in document collection, they are encrypted before outsourcing. These scores are encrypted using order preserving symmetric encryption which maintains order of the scores even after encryption.

The details of secure index construction is given in Algorithm 1.

Algorithm 1 Secure bucket index construction

1: Input: Set of document $C= \{f_1, f_2, f_3 \ldots\ldots\}$
2: Output: Secure buckets
3: Class Mapper
4: Method MAP (totaldoc n)
5: for all documents $f \in C$ do
6: for all term $t \in$ *document f1* do
7: Emit *(term t, score, doc-id)*
8: end for
9: End for
10: end Mapper
11: Class Reducer
12: Method Reducer (term tterm, list)
13: For all term $t \in$ *tterm* do
14: If list already exist for *term t* then
15: Append term to the list of *t* else
16: Create new list for *term i*
17: End If
18: End For
19: Emit $((termt_1, listt_1), (termt_2, listt_2), \ldots\ldots\ldots (termt_{tterm}, list_{tterm}))$
20: End Reducer
21: For $i=1$ *to tterm* do
22: Calculate bucket signature $\pi(B_i) = SHA(t_i)$
23: While (elements in i^{th} bucket) do
24: Vectorbucket $V_{Bt} = (DES (DOC_{id}), OPSE(Score))$
25: End while
26: End for
27: Output secure bucket index I_e

7.2. Secure query generation and searching

Secure query generation is very important. Secure hash algorithm is applied to all the keywords in the query. There are three types of queries supported in this system.

- Single keyword.
- Multi keyword.
- k-NN query.

Let the set of keywords in query $Q = T_q$, the secure query signature is calculated as

$$Sig(T_q) = (SHA (q_1) \ldots\ldots SHA(q_{m2})) \quad (6)$$

There are m_2 buckets required to be accessed for corresponding query signature $Sig (T_q)$. If $m_2 = 1$, it is single keyword search. In this case the signature of queried keyword and bucket identifier is checked. If both matches, top k documents are retrieved from the bucket and given to file server. From there file server pick up those top k documents in the encrypted format from the collection and delivers it to the user. As the keys are already shared between owner and user, user can now decrypt the documents.

Every bucket contains document identifier and relevancy score of data elements which are mapped to that bucket. When multikeyword queries are submitted by users, search server has to calculate document score which is spread across multiple buckets. Suppose given query includes keywords k_1, k_2, k_3. The buckets representing these three features are considered. All the documents present in these buckets are unioned and their respective scores are added to get the final results. If user is interested in top k documents, then k documents having top scores are retrieved. All these top k documents may have all keywords $\{k_1, k_2, k_3\}$ or some keywords. Thus this technique helps not only to retrieve exact matching documents but also partial matching documents.

For multi-keyword logical ORing of all respective m_2 buckets are performed and documents are sorted as per their score. Top k documents are then retrieved as explained above. Similarly, for k-NN query, keywords are extracted from query document, and for all those keywords logical ORing of all respective buckets is

performed as shown in Equation 9. Assume document is having m_2 keywords. Top k documents matching to input query are retrieved as per discussed above.

$$Query(Doc) = \left(B(k_1)OR\ B(k_2)OR\ B(k_m)\right) \quad (7)$$

Details are given in Algorithm 2.

Algorithm 2 Secure query generation and searching

```
Input: Encrypted document collection E(C), query document da, secure index I, top k need
Output: Relevant documents
Initialization: toutput=Null, foutput=NULL, k=1
1: Search Server
2: Tq=Conflate(dq)
3: Secure query generation, T1=(sig(q1), sig(92) ....sig(qm2))
4: Search for all features present in input query document
5: foutput= V Bq1
6:    for i=2 to m2 do
7:        for j=1 to m1 do
8:           if(sig(qi)=sig(tj)) then
9:              toutput= V Btj
10:                while(toutput!=NULL) do
11:                   if(element of toutput is present in foutput ) then
12:                      Update score of doc-id in foutput by adding score from toutput
13:                   else
14:                      append(doc-id, score) from toutput to foutput
15:                end if
16:                end do
17:          end if
18:       end for
19:    end for
20: Retrieve top k elements from foutput
21: Transfer these top k elements to file server
22: File Server
23: Retrieve all top k documents from encrypted collection C
24: Transfer top k documents to user
25: End user
26: Decrypt the retrieval documents by using shared secret
```

8. Distributed indexing

The prime purpose of distributed index is to spread out the work load of an index. Instead of maintaining a single index, distributed index maintains multiple indexes to solve user's request efficiently. The distributed indexing technique increases availability and fault tolerance of the stored index.

Two evident alternatives available are:

- Partition by features;
- Partition by documents.

In partition by features, dictionary of index terms is partitioned into subsets such that each subset resides at a single node. To access the data, each node is attached with the respective posting lists. The master node routes the query to appropriate nodes by using available metadata. In reality, this allows high degree of concurrency as a multi-keyword and k-NN query would hit different nodes. The node in a system need to send appropriate posting lists to a merging node, where the final merging of intermediate results will be done. This result is sorted as per their relevancy score, and top k encrypted document identifiers are dispensed to file server. The two important factors that decides the partitioning are occurrence of terms in the collection and occurrence of terms in the query. The architecture of distributed index on search server is given in **Figure 4**.

The total feature set present in the collection is distributed across various nodes on cloud search server as shown in **Figure 4**. If there are m1 features in the input collection, and n free nodes available on search server then approximately each node is responsible to maintain bucket structure of m1 features.

If any of the node get failed, master node transfers its workload to the next available free node. Thus, it increases the reliability of the system.

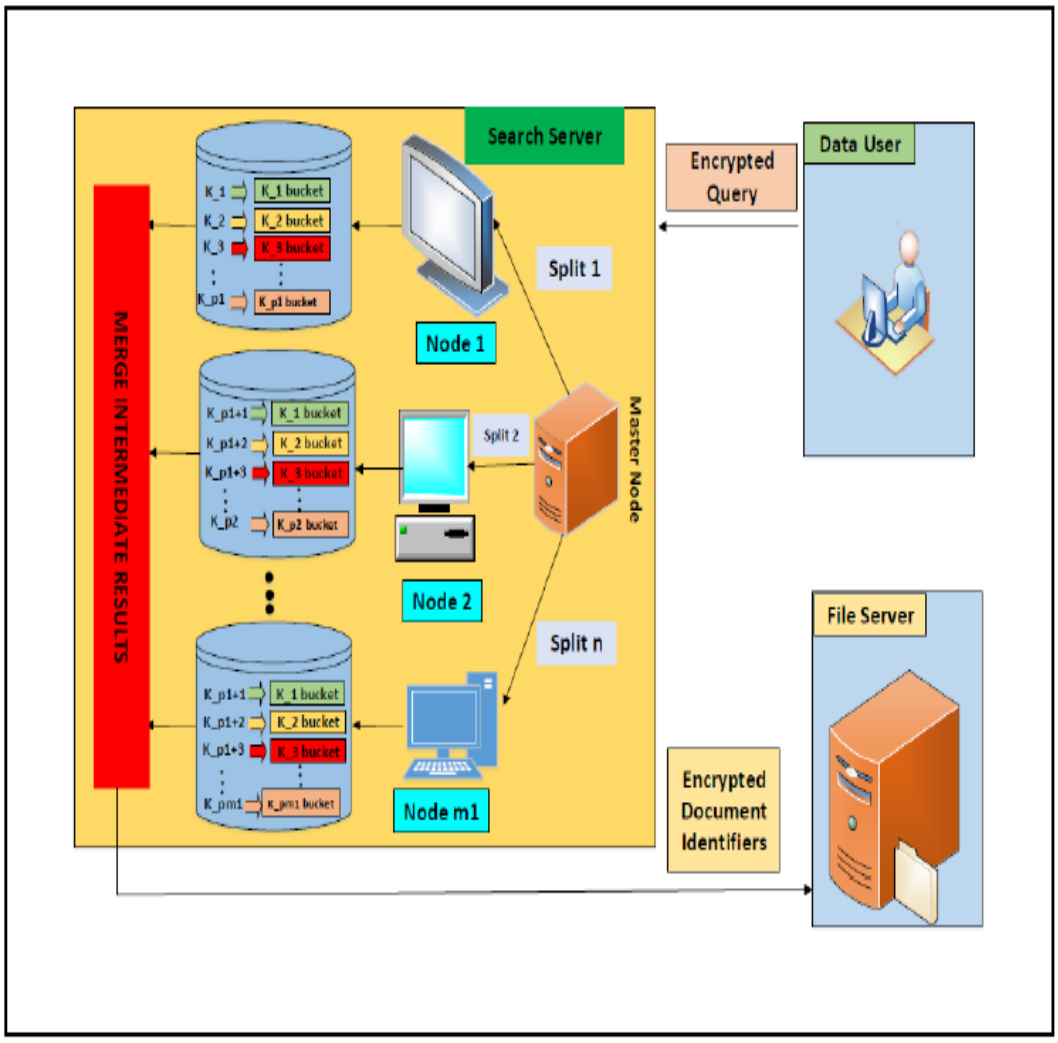

Figure 4. Distributed index.

9. Security analysis

In the proposed system data stored on search server and file server is secure, as it is encrypted by using standard cryptographic techniques. Hence an attacker or adversary will not learn anything about the data stored on both servers. Basically the common way to analyses the security of the system is to formalize the leakage and justify that adversary does not learn anything from this leakage.

Definition 1: Index privacy: The keywords present in the index are secured by using secure hash algorithms. The document identifiers are encrypted; scores are encrypted using order preserving symmetric encryption. Hence system is secure as attacker will not be able infer the index.

$$S(I) = Secure\ (Encrypt(docid), OPSE(score)) \quad (8)$$

Definition 2: Data confidentiality: The collection of data which is stored on file server is encrypted using standard cryptographic algorithm DES, hence it is difficult to infer the data.

Definition 3: Query confidentiality: The query written by end users is secured by using secure hash algorithm. Therefore, even if adversary learn the query or able to infer the query, he cannot extract any information.

10. Performance evaluation

The complete system is implemented using Java programming language. To support the security cryptographic libraries are used under Windows 10 operating system. The execution of proposed system is tested on standard dataset RFC (request for comment). This dataset contains around 8039 text files which can be downloaded[22]. Initially the testing is done on 500 text files having 1835 keywords. The processor is Intel CORE i3.

10.1. Index construction

The very important step in secure data retrieval is index construction. Since the outsourced data is huge in size, the time required to build the index is also more. The proposed method importantly tries to reduce the time for index construction. The total dataset is divided into equal chunks of data. Each chunk of data is processed in parallel by applying map-reduce technique. Therefore, time required to build the index is longest time taken by any of the worker node. Preliminary work is stated in Vasgi, Bharati P et al.[22]. **Figure 5** shows the comparison of proposed method with Wang et al.[23]. From the graph it is clear that parallelizing the index construction reduces time require to build the index. **Table 1** shows the corresponding readings. From the graph it is very clear that time required by proposed method is significantly less as compared to the method proposed in Wang et al.[23]. Crawler based security algorithm is proposed by Wu et al.[24].

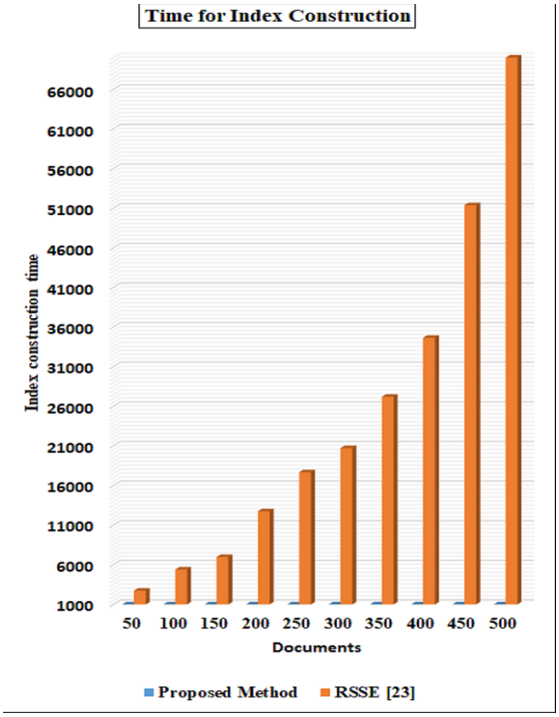

Figure 5. Comparison of index construction time.

Table 1. Index construction time in milliseconds.

Documents	Proposed method	RSSE[23]
50	109	2734
100	145	5406
150	171	6984
200	211	12750
250	248	17681
300	281	20718
350	298	27195

Table 1. (*Continued*).

Documents	Proposed method	RSSE[23]
400	302	34645
450	365	51377
500	404	71414

10.2. Index size

Figure 6 shows the size of index for various files. It is observed that the amount of memory required by secured index is more as compared with original plain-text index. The size of the index is proportional to number of documents. Significant work can be done which will concentrate on index compression techniques. This will definitely useful in pay as you use scenario of cloud computing.

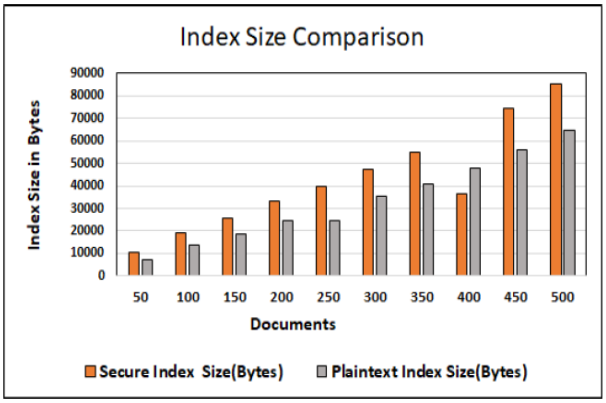

Figure 6. Index size comparison.

10.3. Search efficiency

In outsourcing environment, many times data users are not interested in all documents matching to their request, instead they are interested in top *k* documents. Following are the different ways by which search efficiency of the system is tested.

SHA variations: The signature of bucket is calculated using SHA. We have demonstrated three variations of SHA algorithm as SHA-1, SHA-256 and SHA-512. The time comparison for searching multikeyword query is as shown in **Figure 7**. From **Table 2**, it is clear that as the size of signature increases in length, time required to carry out the search also increases. It indicates that search time is proportional to length of signature. The time cost of SHA-1 is much more efficient than other variations because of length.

Table 2. Multi-keyword search time using SHA variants in milliseconds.

Documents	SHA-1	SHA-256	SHA-512
50	412	422	426
100	473	476	490
150	483	495	502
200	543	550	553
250	512	516	597
300	534	542	587
350	540	552	571
400	547	555	577
450	550	562	580
500	556	565	592

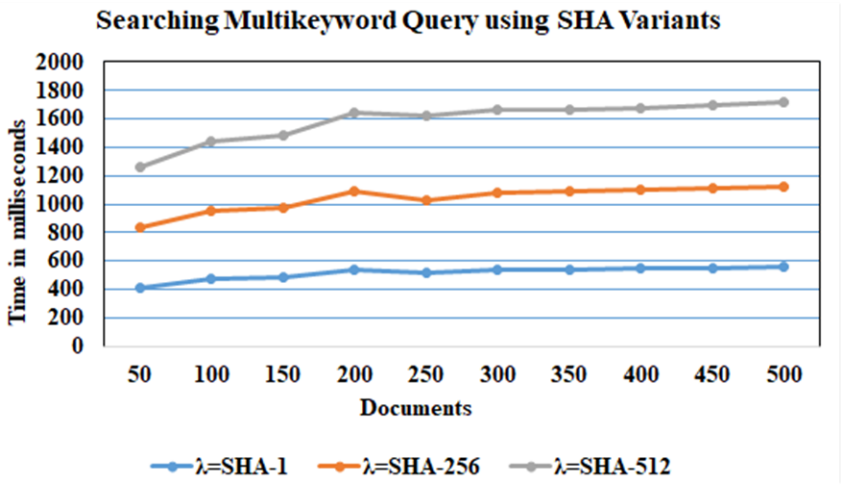

Figure 7. Comparison of multi-keyword search time using SHA variants.

10.4. Search time comparison in secure and unsecured mode

Secure systems always add additional burden on the existing systems. In most of the scenarios where data is very sensitive like healthcare data, government documents or bank databases security is utmost important. The loss or any leakage in this data directly affects organization's revenue and reputation. Hence organizations are ready to bare additional burden imposed by secure system. **Figure 8** shows 4 different cases with secure and unsecured indexes.

- Single keyword and Secure Index.
- Single keyword and Plaintext Index.
- Multiple keyword and Secure Index.
- Multiple keyword and Plaintext Index.

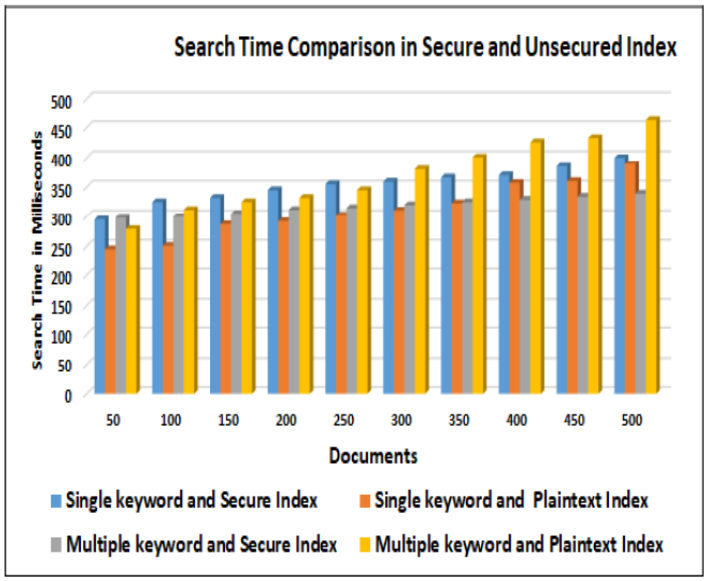

Figure 8. Search time comparison in secure and unsecured index.

From the graph it is very clear that secure systems take more time for searching as compared to unsecured systems. Single keyword and multikeyword searches are shown in **Figure 8**. Many organizations now a days are willing to have this additional burden at the cost security.

10.5. Secure search: Multiple cases

Figure 9 shows graph of 4 types of queries. The queries vary based on number of keywords they are using. From the graph it is clear that time require to carry out search for queries with more keywords becomes comparatively constant. The queries with maximum four keywords are shown in graph.

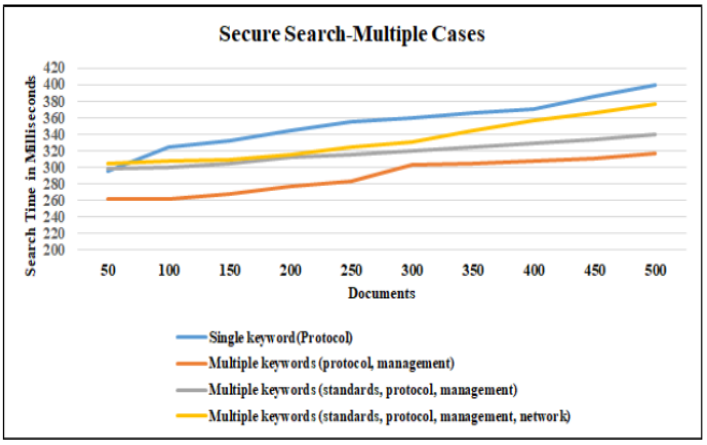

Figure 9. Secure search: Multiple cases.

10.6. Time comparison on search server and file server

Figure 10 shows comparison of time to carry out search on search server and file server.

In order to show the effect of search over the bulk of documents, we tested the scheme for ten sets of 50 to 500 documents. As **Figure 10** shows for 50 documents time on search server is 17 milliseconds and on file server is 85 milliseconds whereas for 500 documents time on search server is 116 milliseconds and on file server is 65 milliseconds. Note that time required on file server is more than that of search server. This is because from file server all relevant files need to be transfer at user's site. Significant work can be done to reduce time on file server. If we observe the proportionality between time required on search server and time required on file server, as number of documents increases this proportionality decreases. It indicates that for larger data sets time on file server decreases. We also observer from **Figure 10** that on average, about 70\% and 30\% of total search time are spent on file server and search server respectively.

Table 3. Search time on search server and file server in milliseconds.

Documents	File Server	Search Server
50	85	17
100	91	27
150	94	31
200	95	30
250	100	34
300	103	37
350	106	43
400	110	44
450	113	49
500	116	65

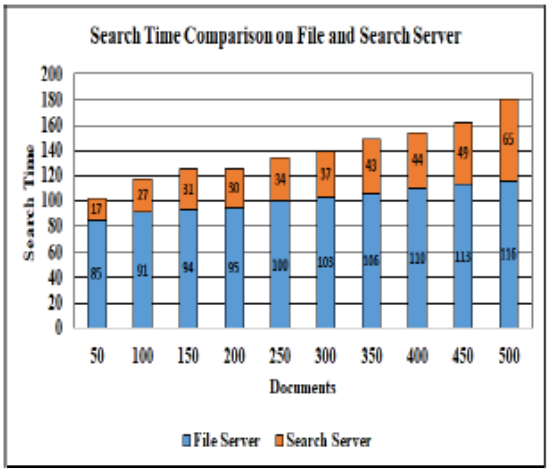

Figure 10. Search time comparison on search server and file server.

10.7. K-NN search

Similar to multikeyword search k-NN approach is also tested on RFC dataset. There is a major difference between multikeyword search and k-NN search. The number of queried terms are much larger than multikeyword search, as all the important keywords from the document are considered in k-NN approach. In case of multikeyword we assume that number of keywords in query are small (e.g. less than 10). **Figure 11** shows search result for document rfc1179.txt.

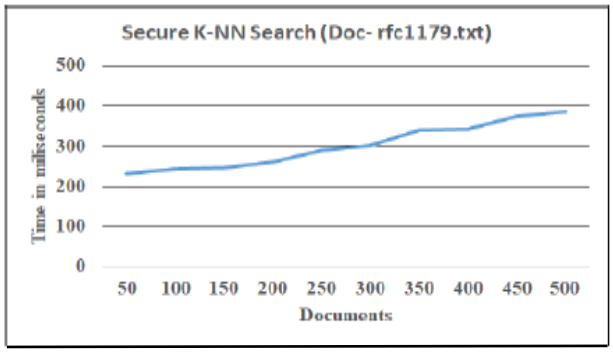

Figure 11. Secure k-NN search.

11. Conclusion

In this paper we propose secure search over encrypted data using parallelization technique.

The proposed system is implemented using map reduce architecture and supports privacy search on large data. Three important variations of secure search are discussed like single keyword, multi-keyword and secure k-NN. The concept of distributed indexing is discussed. The performance of the method is evaluated using standard dataset which are freely available. The result shows that the proposed method is very effective and can be expanded to larger dataset. The future work of the proposed method is to maximize retrieval recall by using query expansion techniques and to reduce size of bucket index with the help of compression methods.

Author contributions

Conceptualization, BPV and SMJ; methodology, BPV; software, BPV and GGC; validation, BPV, SMJ and GGC; formal analysis, BPV; investigation, BPV; resources, BPV; data curation, BPV; writing—original draft preparation, BPV; writing—review and editing, BPV; visualization, BPV, GGC; supervision, BPV; project administration, BPV; funding acquisition, SMJ. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This work is supported by Board of College and University Development, Savitribai Phule Pune University, India (15ENG001320).

Conflict of interest

The authors declare no conflict of interest.

References

1. Yang Y, Li H, Liu W, et al. Secure dynamic searchable symmetric encryption with constant document update cost. In: Proceedings of the 2014 IEEE Global Communications Conference; 08–12 December 2014; Austin, TX, USA. pp. 775–780.
2. Jung T, Mao X, Li XY, et al. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In: Proceedings of the IEEE INFOCOM 2013; 14–19 April 2013; Turin, Italy. pp. 2634–2642.
3. Orencik C, Selcuk A, Savas E, Kantarcioglu M. Multi-Keyword search over encrypted data with scoring and search pattern obfuscation. International Journal of Information Security 2016; 15(3): 251–269. doi: 10.1007/s10207-015-0294-9
4. Strizhov M, Ray I. Multi-keyword similarity search over encrypted cloud data. In: Cuppens-Boulahia N, Cuppens F, Jajodia S, et al. (editors). ICT Systems Security and Privacy Protection, Proceedings of the 29th IFIP TC 11 International Conference (SEC 2014); 2–4 June 2014; Marrakech, Morocco. Springer Berlin, Heidelberg; 2014. Volume 428, pp. 52–65.
5. Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on Parallel and Distributed Systems 2014; 25(1): 222–233. doi: 10.1109/TPDS.2013.45
6. Cash D, Jarecki S, Jutla C, et al. Highly-scalable searchable symmetric encryption with support for Boolean queries. In: Canetti R, Garay JA (editors). Advances in Cryptology—CRYPTO 2013, Proceedings of the 33rd Annual Cryptology Conference; 18–22 August 2013; Santa Barbara, CA, USA. Springer Berlin, Heidelberg; 2013. Volume 8042, pp. 353–373.
7. Chen Z, Wu C, Wang D, Li S. Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor. In: Chau M, Wang GA, Yue WT, Chen H (editors). Intelligence and Security Informatics, Proceedings of the 2012 Pacific-Asia Workshop on Intelligence and Security Informatics (PAISI 2012); 29 May 2012; Kuala Lumpur, Malaysia. Springer Berlin, Heidelberg; 2012. Volume 7299, pp. 176–189.
8. Van Liesdonk P, Sedghi S, Doumen J, et al. Computationally efficient searchable symmetric encryption. In: Jonker W, Petković M (editors). Secure Data Management, Proceedings of the 7th VLDB Workshop (SDM 2010); 17 September 2010; Singapore. Springer Berlin, Heidelberg; 2010. Volume 6358, pp. 87–100.
9. Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on Computer and communications security (CCS 2006); 30–3 November 2006; Alexandria, Virginia, USA. pp. 79–88.
10. Orencik C, Savaş E. An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. Distributed and Parallel Databases 2014; 32(1): 119–160. doi: 10.1007/s10619-013-7123-9
11. Vasgi BP, Kulkarni UV. Data security issues in outsourced environment: A survey. International Journal for Research in Engineering Application and Management (IJREAM) 2018; 3(10): 89–96. doi: 10.18231/2454-9150.2017.0083
12. Vasgi BP, Kulkarni UV. A secure and effective retrieval using hash-based mapping structure over encrypted cloud data. International Journal of Electrical Electronics and Computer Science Engineering 2017; 4(4): 65–74.
13. Wong WK, Cheung DWL, Kao B, Mamoulis N. Secure kNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD 2009); 29–2 July 2009; Rhode Island, USA. pp. 139–152.
14. Shi E, Bethencourt J, Chan TH, et al. Multi-dimensional range query over encrypted data. Security and Privacy. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP 2007); 20–23 May 2007; Oakland, California, USA. pp. 350–364.
15. Hore B, Mehrotra S, Canim M, Kantarcioglu M. Secure multidimensional range queries over outsourced data. The VLDB Journal 2012; 21(3): 333–358. doi: 10.1007/s00778-011-0245-7
16. Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval, 5th ed. Addison Wesley; 1999.
17. Boldyreva A, Chenette N, Lee Y, O'neill A. Order-preserving symmetric encryption. In: Joux A (editor). Advances in Cryptology—EUROCRYPT 2009, Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques; 26–30 April 2009; Cologne, Germany. Springer Berlin, Heidelberg; 2009. Volume 5479, pp. 224–241.
18. Stallings W. Cryptography and Network Security: Principles and Practices, 7th ed. Pearson; 2016.

19. Katal A, Wazid M, Goudar RH. Big data: Issues, challenges, tools and good practices. In: Proceedings of the 2013 Sixth International Conference on Contemporary Computing (IC3); 8–10 August 2013; Noida, India. pp. 404–409.
20. Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data. In: Vadhan SP (editor). Theory of Cryptography, Proceedings of the 4th Theory of Cryptography Conference (TCC 2007); 21–24 February 2007; Amsterdam, The Netherlands. Springer Berlin, Heidelberg; 2007. pp. 535–554.
21. Manning CD, Raghavan P. Introduction to Information Retrieval. Cambridge University Press; 2008.
22. Vasgi, Bharati P., Girija G. Chiddarwar, and S. M. Jaybhaye. "Novel Frequency Based Natural Language Query Search in Cloud Computing." Computer Integrated Manufacturing Systems 29, no. 5 (2023): 1-5.
23. Wang C, Cao N, Ren K, Lou W. Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Transactions on parallel and distributed systems 2012; 23(8): 1467–1479. doi: 10.1109/TPDS.2011.282
24. Wu X, Wei D, Bharati P, et al. Research on Network Security Situational Awareness Based on Crawler Algorithm. Security and Communication Networks 2022; (2022).