A multi keyword top k search schema for efficient query processing over encrypted data

M. Dhanushaasri[1] , M. Gajendran[2] , C. Hari Varshini[3] , S. Senthamil[4], S. Yuvalatha[5]

UG Scholar [1, 2, 3, 4], Assistant Professor [5]
Department of CSE, Sree Sakthi Engineering College, Coimbatore

Abstract

Cloud computing deliver computing resources, services and virtual management applications over web. Such applications contain sensitive information and the data must be secured from the storage unauthorized third parties and cloud service providers. The privacy and respectability of the information needs to protected during the capacity and transmission. Before the information is redistributed to the servers the information must be encoded utilizing a solid security saving encryption algorithm that must likewise strengthens effective looking over scrambled information. So, to encrypt the information the symmetric AES encryption technique is utilized in applications like HTTP and OFTP.. The cloud server practices an index to search the encrypted data deposited in it to complete the request from the client which is built using the Random traversal algorithm. During the information retrieval, the cloud server uses the Random Group Multi keyword Top – k Search scheme to calculate the relevant scores of each document in the index and display the results to the user. In Random Group Multi keyword top – k Search algorithm the framework overwhelmed the problem of query unlink skill and hiding the access patterns from the cloud server.

Index Terms— *cloud computing, secure encryption and retrieval, privacy preserving, top-k search*

I.INTRODUCTION[1]

Cloud computing is used to deliver computing resources to the users from anywhere at any time through their associated devices. The significant features of cloud computing that consist of high scalability and reliability and pay as per you use approach has focused more users to adapt the cloud computing.

Presently a-days the greater part of the associations even little and medium undertakings convey their applications in cloud because of diminished equipment and support cost, expanded versatility and burden adjusting components.

Yet at the same time a significant number of the associations have a dread to adjust cloud in light of the fact that their information comprises of touchy data. Aside from the information being shared there is a need to realize what information the cloud service provider gathers and security of the information that is put away in the cloud [1].

The confidentiality and integrity is to be ensured during the data storage and transmission in the cloud. To achieve confidentiality the access control needs to be implemented as it consists of both authentication and authorization.

The encryption algorithm is also considered during the design process of the cloud data storage and information retrieval. We use the symmetric encryption algorithm to encrypt the data in the server [2].

The traditional keyword search mechanisms are only applicable to search unencrypted data in the server and efficiency is failed when used in the big data applications [3]. These applications are not efficient in retrieving the encrypted data in the cloud servers.

But in this development we need to search over the encrypted data which is encrypted using the symmetric AES algorithm [4][5].

The multi keyword top-k search schema is used to search in the big data applications and return the top-k documents with the highest significance scores [6]. To calculate the related score of each document the term frequency x Inverse document frequency (tf x idf) model is used[7].

II. PROBLEM FORMULATION

A. Architectural model

. In our framework there are three entities in the structure as follows, the data owner, The cloud service provider and the data user. The data owner outsources the data to the cloud server.

Before the redistributing is done the information is encoded utilizing the symmetric AES calculation. At the point when the information is to be gotten, the server needs to look through the encrypted information put away and so as to do this effectively

The data owner itself builds a searchable index using the random traversal algorithm and outsources it to the server. Then both the data and the searchable index is delegated to the server

The information proprietor itself makes an accessible record utilizing the unbalanced traversal calculation and reorganizes it to the server. At that point both the information and the available file is re-appropriated to the server

When the data user needs to access the data, authorization is done to ensure whether third party is trying to access the server. The permission is done using the two-way authentication mechanism to ensure high security for the users.

After it`s done the user can search the documents in the server using multiple keywords. The cloud server then computes the related score of each document stored in the index using the [TF x IDF] model[8][9]. With the help of the RGMTS algorithm the top-k documents for the given query is returned to the data users.

For the decryption of the data the user receives the key to decrypt from the data owner through a secure channel.

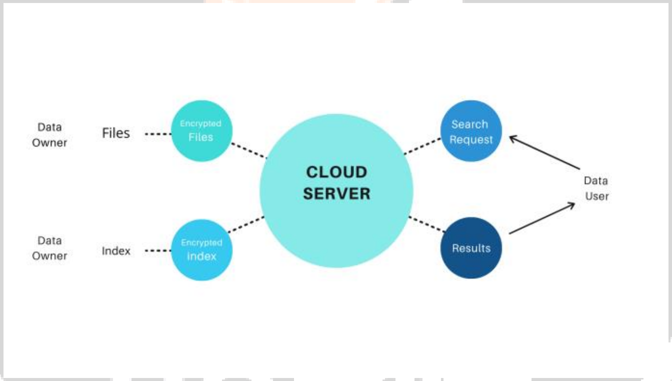

Fig. 1. Architectural Model and Work Flow

B. Threat model

In our system, we think about the information proprietor and the information client as the confided in entities. Be that as it may, we have to guarantee whether the information is being utilized by the cloud server. The cloud server can examine the list and the touchy information that is being put away. So we need a safe encryption and proficient recovery calculation to process the inquiry from the information client.

C. Design goals

1. Encryption of data

The data has to be strongly encrypted and redistributes to the cloud server using a strong encryption mechanism. The encryption mechanism has to upkeep multi-keyword top-k similarity search over encrypted data.

2. Efficient information retrieval

Single keyword search techniques cannot deliver high search competence when the data is encrypted with a strong mechanism.

In big data applications these search techniques are inefficient and incapable [10]. In order to overcome this, we need both secure and efficient query processing and searching mechanism which needs high integration with big data applications.

3. Other issues

The query processing algorithm should shroud the meeting ways and the entrance designs from the cloud server. Since the entrance examples may uncover the data about the archives or the information that is being put away in the server.

The plain content data about the redistributed information and the record must be kept covertly by the information proprietor. Be that as it may, for this situation the information proprietor is considered as a confided in element.

III. RELATED WORKS

With the character of low upkeep, distributed computing gives a prudent and productive answer for sharing gathering asset among cloud clients. Sadly, sharing information in a multi-proprietor way while protecting information and personality security from an untrusted cloud is as yet a difficult issue. Accessible symmetric encryption (SSE) permits a gathering to redistribute the capacity of its information to another gathering (a server) in a private way, while keeping up the capacity to specifically look over it[13]. Most traditional accessible encryption plans experience the ill effects of two burdens. To begin with, looking through the put away records requires some serious energy direct in the size of the database, as well as utilizations substantial number juggling operations[14][15]. Besides, the current plans don't think about versatile aggressors; a hunt question will uncover data even about archives put away later on.

IV. RANDOM TRAVERSAL ALGORITHM

The random traversal algorithm brands server to randomly traverse the index while handling a query to provide dissimilar search results for identical queries by maintaining the accuracy of the queries with enhanced security.

A. Enlarge the document collection

The multiple documents in the collection are randomly distributed into L groups with each group of same size. the collection is represented as follows,

DG = { DG1,DG2,...,DGL }

The each document group is copied E times and also assigned with a unique identifier to each document.

The enlarged document collection is represented as follows,

$DGx = \{DG_1^1,DG_1^2,DG_2^1,DG_2^2\}$

$=\{\{D_1^1,D12\},\{D_2^2,D_1^2\},\{D_3^1,D_3^4\}\{D_4^2,D_3^2\}\}$

$=\{D_1^1,D_2^1,D_2^2,D_1^2,D_3^1,D_4^1,D_4^2,D_3^2\}$

B. Assign switch

The document in the group is allotted with a switch value, r. The switch is a vector value which is a multiple of L and E, where the no of groups is the document collection has been divided and E is the no of times the document has been imitative.

C. Build index

We need to construct a tree based index (I) for the whole document collection. Let N represents a node in I, and it is denoted as { fid, lc, rc, switch}. If N is a leaf node, fid is the document identifier, lc and rc are null. Else fid is null, lc and rc point to its left and right child.

D. Assigning key

The key value and length is equivalent to the switch esteem. The question is doled out with various key qualities to get diverse query items and visiting ways. While producing a key, information client chooses one measurement from every E measurements of key, and the chose measurements are set to zero, while the others are set to various arbitrary negative whole numbers on the grounds that there are E duplicates for each report and the cloud server needs to traverse just one duplicate of the record without fail.

E. Query processing

The query processing begins with the root hub of the list. In the event that the worth is negative for the hub, at that point the server won't visit it's kid hubs. It navigate through each hub in the record until the worth is certain or zero to every hub.

V. GROUP MULTI KEYWORD TOP-K SEARCH SCHEME

The group multi keyword top-k search scheme is utilized to recover archives from the scrambled information. It partitions the catch phrase in the word reference into various gatherings and makes an accessible record for each gathering. The server restores the archive with the most elevated applicable score i.e., it restores the top-k records for every watchword to the client. The systems think about different GMTS plans and consolidate the arbitrary traversal calculation and the irregular gathering multi-catchphrase top-k search plan to improve productive outcomes to the information client.

UGMTS [unencrypted group multi-keyword top-k search scheme]
EGMTS [encrypted group multi-keyword top-k search scheme]
RGMTS [random group multi-keyword top-k search scheme]

A. Unencrypted Group Multi-keyword top-k search

The data owner has to build the searchable index for the document collection locally. The index consists of IC and IR. The IC is used to select the documents and IR is used to calculate the relevant score of each document.

1) Building index

The inverted index (V) is created that consists of inverted lists and it is denoted as follows,
{vl(w1), vl(w2), .., vl(wn)}.
Then dictionary is divided into multiple groups
W G = {W G1, W G2, ..., W Gb}

Each group contain d keywords. The data owner also finds the top *c _ k* documents of each word group based on the inverted index *V* , denoted V G = {V G1, V G2, .., V Gb} where *V Gi* is the top *c _ k* documents of word group *WGi*.

Then a keyword balanced binary tree *ICi* is built as an index for each keyword group *WGi*.
The indexes combine as IC = {IC1, IC2, ..., ICb}
Then another index IR is built for IR = {IR1, IR2, ..., IRm}

2) Query Construction

When the data user wants to search with keyword set *Wq*, they generates query group *Q*. The query group is represented as Q = {QC, QR}where *QC* is used to search on index group *IC* and *QR* will be processed in *IR*.

Query group *QC* is denoted as {QC1, QC2, ..., QCb} represents a query in *QC* and its a query vector with length *d*.

The other query group *QR* is denoted QR = {QR1, QR2, QRb} and it is the same as *QC* in UGMTS. The data user submits query *Q* to the cloud server.

3) Query Processing

The query Q is received by the cloud server and it processes the QC on the IC to retrieve the documents in the CList. The relevance scores between *QCi* and the nodes of *ICi* are calculated
Score(QCi , Ni) = (QCi) · (Ni .val)
Then the final relevant score is calculated between QR and the documents in the IR using
Score(QR, IRi) = ∑ QRj∈QR (QRj) · (IRi .valj)

B. Encrypted Group Multi-keyword top-k search

In EGMTS , to protect the real value of indexes and queries some random values are added to the real relevant score and this value is encrypted by the secure KNN algorithm. The EGMTS algorithm is constructed as follows,
The data owner generate to keys *sk*1 and *sk*2. The values are as follows *sk*1= {S1, M1, M2}and *sk*2 = {S2, M3, M4}
The S1 contains (d+u+1) vectors denoted as follows,
{S_1^1 , S_2^1 , ..., S_b^1} for decryption through a secure channel.

1) Build index

a) Magnify values of IC

In UGMTS the values of data vectors in intermediate is the max value of its children and the cloud server can identify this by linear equations. So, the values are magnified by adding random numbers. val[j] = max{lc.val[j], rc.val[j]} + |rand()|%max{lc.val[j], rc.val[j]}

b) Extending IC

The dimension of each data vector in *IC* is extended from *d* to *d*+*u*+1, and *u* is the number of phantom terms. The values of phantom terms are randomly set to 0 or 1

c) Extending IR

Each vector in the *IR* is extended from *d* to (*d*+ *r*) and a vector of length (*r* + *u* + 1) is added to each index. The values of extended dimensions and phantom terms are set to
0 or 1, and all the (*r* + *u* + 1)-th dimension of added
vectors are set to 1.

d) Encrypt index

The node Ni is used to represent a node in the index ICi and *S*1*;i* to denote the *i*-th vector in *S*1. The data owner splits vector *NVi* into two random vectors {NV'_i , NV''_i } {$M^T_{1,i}NV'_i$, $M^T_{2,i}NV''_i$ } After splitting process is complete, node *Ni* stores two {$M^T_{1,i}NV'_i$, $M^T_{2,i}NV''_i$ } encrypted vectors where *M*1*;i* and *M*2*;i* represent the *i*-th matrices in the matrix groups *M*1and *M*2, respectively.
The data owner also encrypts *IR* with secret key *sk*2,
where the encryption method is the same as encrypting *IC*.

2) 3)Build Query
The method of generating query groups *QC* and *QR* is similar to the UGMTS. But some random and phantom values are added to extend the values of QC and QR.

a) Extending QC

The query vectors in *QC* are extended from *d* to *d* + *u* + 1 dimensions. The values of phantom terms are set to random numbers _*i;j* and the (*d* + *u* + 1)-th dimension is set to another random number λi.

b) Extending QR

The data users add phantom value (d + u + 1) is added to the QR. Then r dummy keywords are added to each query and their summation value is set to zero. This helps to restrict server from tracking the relevant score between the document and the keyword in the query.
The SK2 is used to encrypt QR as the same as QC.
3) Query processing
The query processing steps are the same as in the UGMTS. The relevant score differs from the UGMTS as follows,
$\mathrm{Score}(QC_i,N_i)= \gamma\ (\mathrm{score}(QC_{,,},N_i)+\sum_{j=1}^{u} \varepsilon_{i,j}+ \lambda_i$

C. Random Group Multi-keyword Top-k search scheme

1) Building index
The data owner enlarges the document collection
D to *DGx* and a random switch is assigned to each document.
The keywords in the dictionary W is divided into same groups of similar size and each top-k document associated with the keyword is identified. Then the *V Gi* is extended to *V Cx. V Gi* is the top-*ck* documents of keyword
group *WGi*, and *V Cxi* is a subset of *DGx* w)

1. The query vector in *QC* is extended from (*d* + *u* + 1) to *e*;
2. Each query of *QC* is assigned a random key
3. The data user assigns a random key to the phantom query in the query group *QR*.

The data user submits the query to the cloud server using the trapdoor T. The cloud server then calculate the relevant score between the Qc and Ic. It traverse the index tree from the root node to the leaf node if and only if the score of intermediate node is larger than zero. If the values are negative then it`s children nodes will not be traversed.

Then the documents having the highest relevant score will be given to the data user.

VI. PERFORMANCE ANALYSIS

In this section the performance of our Group Multikeyword Top-k Search algorithms are analyzed.

A. Index construction

When building the IC each index stores the top-ck documents which generates $O(\Omega mb)$ nodes. Let us calculate the time to know the actual time need to encrypt the index.
The node encryption techniques involve two processes i.e., splitting process and multiplication of two e matrices. the dictionary splitting process takes $O(d)$ time and the two multiplications takes $O(e2)$ time. We then compare our framework with the EDMRS method to know the efficiency of our framework [11]. The time complexity of EDMRS is $O(n2m)$. In EDMRS the node is encrypted by the value 0f n where n=|w| w is the size of dictionary. The node encryption takes place by encryption of n x n matrices which needs more time. The time complexity of index construction is approximately equal to O ($\_mbe2$). The RGMTS consumes less time to encrypt the index than the EDMRS algorithm.

B. Trapdoor generation

The length of trapdoor is equivalent to the size of the word reference in the EDMRS component. At the point when the information client looks through the information utilizing catchphrases not as much as 'n' it needs additional time and processing power. In any case, in the GMTS, the trapdoor is isolated into b parts and each part is a question with length 'd'. On the off chance that the 'd' measurements of the vector is zero then the question i.e., the part is expelled from the trapdoor [12]. Along these lines it requires some investment to scramble the hubs in the list. The time intricacy for trapdoor development is O(te2). The trapdoor age time is influenced by the quantity of questions and the size of word reference

C. Search

We previously expressed that the inquiry given by the client is separated into different question gatherings and just the non-void question bunches are send to the server. In this manner the server doesn't have to look through all the record of all catchphrase gatherings. In the event that the important score between the hub and the catchphrase in the inquiry is zero or negative then the hub and its kids won't be crossed. This assists with crossing all the hubs in the record which has pertinent score higher than zero for the given inquiry. Likewise, the computational expense of the system is diminished somewhat.

VII. CONCLUSION

In this paper we essentially center around shielding the delicate data from outsider and productive data recovery over the encrypted information. From the start the symmetric encryption is use to scramble the information yet we our objective significantly means to effective information recovery over encoded information. Then the index is encrypted using the random traversal algorithm (RTRA) and the data is outsourced to the cloud. The Random Group multi keyword top-k search algorithm along with the RTRA is used for search and information retrieval which helps to hide the access patterns and the visiting paths from the cloud server that preserves the privacy of the data. The framework helps protects the data from the server to access the data using the linear attacks.

The performance analysis show that the framework provides requires less execution time and computing resources for the execution of the algorithms.

VIII. REFERENCES

[1] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," ACM Computing Surveys, 2016

[2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Advances in CryptologyEurocrypt 2004. Springer, 2004, pp. 506–522 .

[3] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in Information and Communications Security. Springer, 2005, pp. 414–426

[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Security and Privacy, 2000. SP 2000. Proceedings. 2000 IEEE Symposium on, 2000, pp. 44–55

[5] E.-J. Goh et al., "Secure indexes." IACR Cryptology ePrint Archive, vol. 2003, p. 216, 2003

[6]W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in Proceedings of the 8th ACM SIGSAC Symposium on Information, ser. ASIA CCS '13. ACM, 2013, pp. 71–82

[7]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

[8]C. D. Manning, P. Raghavan, H. Schutze ¨ et al., Introduction to information retrieval. Cambridge university press Cambridge, 2008, vol. 1, no. 1

[9]S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," Computer Networks and ISDN Systems, vol. 30, no. 17, 1998

[10]W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. ACM, 2009, pp. 139–152

[11]Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 340–352, 2016.

[12]J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in INFOCOM, 2010 Proceedings IEEE, 2010, pp. 1–5.

[13]Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in Applied Cryptography and Network Security. Springer, 2005, pp. 442–455. C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May

[14] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in Information security applications. Springer, 2004, pp. 7386

[15] X. Yuan, H. Cui, X. Wang, and C. Wang, "Enabling privacy similarity retrieval over millions of encrypted records," in European Symposium on Research in Computer Security. Springer, 2015, pp. 40–60.