ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Efficient Multi-Keyword Ranked Query Scheme over Encrypted data in Cloud Computing

Shaik.Asifa
M.Tech Student,
Audisankara College of Engineering and Technology, Gudur.

Prof. Rajendra. Chadalawada
Professor & HOD,
Department of CSE,
Audisankara College of Engineering and Technology, Gudur.

Dr.A. Sri Lakshmi
Lecturer in Computer Application,
Govt. Degree College (w),
Srikalahasti, Chittoor Dt.

ABSTRACT:

Cloud computing provides an improvement of expansive scale data storage, processing and distribution. But for protecting data privacy is a major concern .This makes to support efficient keyword, based queries and rank the coordinating results on the encrypted data. The existing work considers Boolean keyword search without appropriate ranking schemes. In the current multi-keyword ranked search approach, the keyword dictionary is static and can't be developed effectively when the number of the keywords increases. Moreover, it doesn't consider the client search keyword access frequency into account. For the query coordinating result which contains a large number of documents, the out-of-order ranking problem may happen. In this paper, we propose a multi-keyword query scheme over Encrypteddata called MKQE to address the previously stated drawback. MKQE greatly reduces the maintenance overhead during the keyword dictionary expansion. It takes keyword weights and client access history into thought when generating the query result.

KEYWORDS:

Cloud Computing, Searchable Encryption, Keyword Search, Ranked Search.

I.INTRODUCTION:

Cloud computing is getting more consideration from both scholastic and industry groups as it turns into a organization stage of deployment platform of distributed applications, particularly for large-scale information management systems.

End clients can outsource their own information onto public clouds, and afterward get to the information at whatever time and anyplace. Cloud computing infrastructure gives an adaptable and monetary system for data management and resource sharing. It can decrease equipment, software costs and framework maintenance overheads. With the fame of cloud services, for example, Amazon Web Services Microsoft1, Azure2, Apple iCloud3, Google AppEngine4, more organizations want to move their data onto the cloud. for giving the security encrypt private data before transferring it onto the cloud server. On one hand, this methodology guarantees that the information are not visible to outside clients and cloud administrators.

There are extreme handling confinements on encrypted data. For instance, standard plain content based searching algorithms are not pertinent any more. To perform an keyword based inquiry, the whole data set must be decrypted regardless of the fact that the coordinating result set is little. To understand this issue, current solutions use the following strategy to provide keyword-based searching capabilities on encrypted data. to meet the effective data recovery require, the expansive measure of document demand the cloud server to perform result relevance ranking, rather than returning undifferentiated results. For security insurance, such ranking operation, on the other hand, ought not to release any keyword related data.

To enhance the query search result accuracy as well as to enhance the user searching experience, it is additionally fundamental for such ranking framework to support multi keywords search, as single keyword search regularly yields unreasonably coarse results.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

II.RELATED WORK:

The issue of Private Information Retrieval was initially presented by Chor et al. [1]. As of late Groth et al.propose a multi-inquiry PIR technique with steady correspondence rate. Be that as it may, any PIR-based strategy requires highly costly cryptographic operations keeping in mind the end goal to hide the access pattern.. A late work proposed by Wang et al. [2] permits ranked search over an encrypted database by using inner product similarity.. Be that as it may, this work is just constrained to single keyword search queries.et al. [2] permits ranked search over an encrypted database by using inner product similarity. Nonetheless, this work is just constrained to single keyword search queries.

Single Keyword Searchable Encryption: Traditional single keyword searchable encryption conspires as a rule constructs an encrypted searchable index such that its substance is covered up to the server unless it is given suitable trapdoors created by means of secret key(s) [3]. It is initially mulled over by Song et al. [4] in the symmetric key setting, and upgrades and propelled security definitions are given in Goh [5], Chang et al. [6], and Curtmola et al. Our initial works [8], [9] solve secure ranked keyword search which uses essential word recurrence to rank results as opposed to returning undifferentiated results. Be that as it may, they just supports single keyword search. In general public key setting, Boneh et al. [7] present the first searchable encryption development, where anybody with public key can keep in touch with the data put away on server yet just approved clients with private key can search.

Boolean Keyword Searchable Encryption: To enhance search functionalities, conjunctive keyword search over encrypted data have been proposed. These plans acquire expansive overhead brought about by their central primitives, such as computation expense by bilinear map, for instance, [11], or communication cost by secret sharing, for instance, [10]. As a more general search methodology, predicate encryption schemes [12] are as of late proposed to support both conjunctive and disjunctive search. Conjunctive keyword search gives back “all-or nothing,” which implies it just returns those records in which all the keywords determined by the search query appear; disjunctive that keyword search returns undifferentiated results, which implies it gives back each document that contains a subset of the particular essential keywords, even stand out keyword of interest.

III.PROBLEM FORMULATION

MRSE Algorithm:

1. For the first time, we explore the issue of multi keyword ranked search over encrypted cloud data, and build up an arrangement of strict protection necessities for such a safe cloud data usage framework.

2. We propose two MRSE schemes in light of the likeness measure of “coordinate matching” while meeting diverse security necessities in two distinctive threat models.

3. We examine some further upgrades of our positioned pursuit component to support more search semantics and dynamic data operations.

4. Intensive examination researching security and productivity assurances of the proposed plans is given, and tests on this present real-world data set further demonstrate the proposed plans in fact present low overhead on calculation and communication.

Limititations of MRSE:

This methodology experiences three major disadvantages. To begin with, it utilizes a static dictionary. In the event that new keywords to be included, the dictionary must be rebuild totally this prompts considerable computational overhead. Second, an out-of-order issue happens if utilizing its trapdoor generation algorithm. Such an issue carries the outcome that the files with all the more coordinating keywords are likely excluded from the top k positions in the coordinating set. This implies that the data consumer will be unable to locate the most significant files they need. In conclusion, MRSE does not consider the impacts of keyword weight and access frequencies. In this way the the files that contain frequent keyword may not be incorporated in the top k areas in the returning result by any means.

IV.PROPOSED SOLUTION:

A.System Model:

Considering a cloud data facilitating administration including four distinct substances, as represented in Fig. 1: the administrator, the data owner (DO), the data client or Data consumer (DC), and the cloud service provider (CSP).

The administrator will deal with the production of accounts for data owner and verifying of data user qualifications. The data owner has a collection of data documents F to be outsourced to the cloud server in the encrypted structure C. To empower the searching capability over C for effective data utilization, the data owner, before outsourcing, will first form a encrypted searchable index I from F, and after that outsource both the index I and the encrypted document collection C to the cloud server. To search the document collection for t given keywords, an authorized users a procures a corresponding trapdoor T through search control mechanisms.

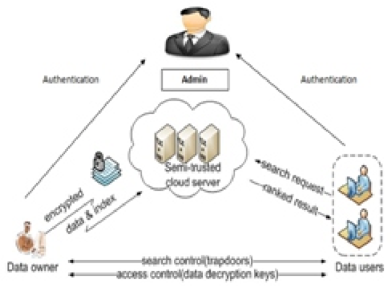

Fig. 1. Architecture of the search over encrypted cloud data.

B.Notations:

• F : the set of original files, assuming there are m files. F is denoted as F = (F1, F2, F3. . . Fm).
• C: the set of encrypted files, corresponding to the files in F . C is denoted as
 C = (C1, C2, C3. . . Cm).
• W: keyword dictionary, assuming we have n keywords. W is denoted as
 W = (W1,W2,W3. . .Wn).
• Fid: the index set of every file, which is denoted as Fid = (Fidx1, Fidx2, Fidx3. . . Fidxm).
• indexx: the keyword set of each Findx, assuming it contains keywords indexkw1,kw2...kwk.
• weightx: the weight set for keywords in index indexx, weightx = (w1,w2,w3. . .wk).
• p: the index vectors for Fidx. p is denoted as p = (p1, p2, p3 . . . pn).
• I: the encrypted index vectors for p. I is denoted as I = (I1, I2, I3 . . . In).
• Wq: a plain text query, assuming it contains k keywords, and can be represented as Wkw1,kw2...kwk.
• weightq: the weight set for keywords in query Wq, weightq =wq1,wq2. . .wqk.
• q: for a query Wq ,the corresponding query vector.
• T : the trapdoor for a query Wq, which is based on q.
• R: the list of files in the returned matching result set. It is a sorted list, the order of the files is determined by the scores.

C.weighted keyword query:

At the point when producing the query result, considering the keyword weights is important, particularly for multi-keyword query situations. Commonly, if the quantity of files containing certain keywords is large, selecting a suitable subset which best coordinating the DC's necessity gets to be critical. In this manner, therefore, the query results should be ranked. Certain measurements must be utilized to create this order and reflect the DC's need. Weighted inquiry issue has been very much concentrated on in the past examination lives up to expectations. Arrangements, for example, the term-weighting [14] or similitude space [15] query are being proposed. Be that as it may, every one of these works just considers the queries on the plain content documents. On the other hand, in our situation, both the files and the keyword indexes stored on the cloud servers are encrypted. Subsequently, the CSP can't have any significant bearing those systems to lead the query directly .New algorithms have to be developed. In this paper, a practical and novel keyword query framework on encrypted data is developed. System design we propose MKQE, another answer for location the above issues. MKQE incorporates an arrangement of novel techniques to give successful and effective mechanisms on the issue of the multi keyword ranked query on encrypted data. In this area, we display the points of interest of our solution.

D.MKQE Framework:

The MKQE system consists of the following components:
• Setup: based on the sensitive data, the DO determines the keyword dictionary size n, the number of dummy keywords u, and then sets the parameter d = 1 + u + n.
• Keygen(d): the DO generates a secret key SK k1, two invertible matricesM1 andM2 with the dimension d×d, and a d-bit vector .
• Extended-Keygen(k1, z): if z new keywords are added in the dictionary, the DO generates a new SK k2, two invertible matricesM′1andM′2 with the dimension d+z×d+z, and a new (d+z)-bit vector S′ .

• Build-Index(F,SK): for each file, the DO determines the set of keywords Fidx, and builds p for it. Then it encrypts the index vector with an SK (either k1 or k2). After that, all the encrypted indexes are added to I. All the files are encrypted with DES or AES, and added to C. Finally, upload I and C onto the CSP.
• Trapdoor (Wq, SK): The DC sends a multi-keyword ranked query Wq to the DO. The DO generates an index vector q and calculates the trapdoor T using an SK and sends it back to the DC.
• Query (T, k, I): The query is sent to the CSP. The CSP runs the query on I and returns the most relevant top k scored files back to the DC.

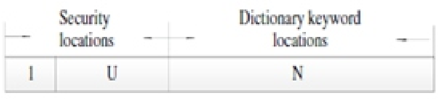

Fig. 2.The index and trapdoor structure in MKQE

E.Extended Key Generation:

As we examined in past areas, a secret key can be utilized to encrypt file index vectors and query vectors. Be that as it may, in MRSE, the size of the matrices and the vector in the secret key is determined by the keyword dictionary size n. In the event that the dictionary is extended, the estimation of n needs to change. To determine this issue, in MKQE, we present another methodology utilizing partitioned matrix operations to lessen the computational overhead. In MKQE, when there are new keywords included the dictionary we don't change the first secret key. Rather, we just add another secret key to support queries for the newly added keywords.

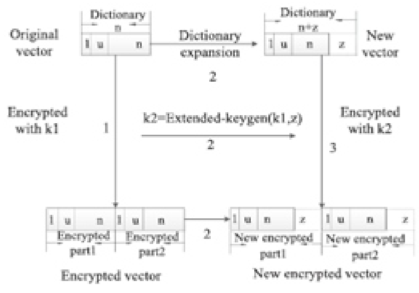

Fig. 3.Keyword dictionary expansion operations

F.Index Building:

A major advantage of MKQE is that we give the easy keyword expansion capacity with low overheads.

In MKQE, we accept that the arrangement of keywords could change every once in a while. In such a situation, just minor changes are required for the first arrangement of files. For a certain file, MKQE extends the corresponding index vector by adding z element at last. The values for this recently included element are resolved as takes after. In the event that it contains a recently included keyword j ($1 \leq j \leq z$), then in its relating index vector, p[1+u+n+j] is set to 1, else it is 0. Clearly, the initial 1 + u + n areas are unaltered. In MKQE, new inverse matrices can be put away utilizing Encryption-Algorithm. After that, we manufacture the index vector p utilizing the keywords as a part of the index. The procedure is the same as MRSE. Other than that, in MKQE, we develop two additional vectors p1 and p2 utilizing formulas (1) and (2). Plain text vector (p1 or p2) are utilized as the parameter vector to be encrypted in the Encryption-Algorithm.

Linear Genetic algorithm adopts linear recursive method as shown below:

After generation 1, the numbers of the next generation is obtained by CROSSOVER followed by MUTATION.
The pairing up of numbers is done first, with the concept that for odd type generation pairing is done in one way and
for even type generation in the opposite way. For example, after the first generation we got the following numbers: - 333, 6578, 8614, 5959, 7922, 8837, 4440, 903, 3693, 2686.
2nd Generation: -Pairing up: - (333, 6578), (8614, 5959), (7922, 8837),(4440, 903), (3693, 2686)For this generation crossover and mutation will take place let at 6th locus of the gene on chromosome.
CROSSOVER: -
Binary Representation of the first pair:
333 = 0000101001101
6578 = 1100110110010
Crossover:
0000100110010 1100111001101
MUTATION: -
0000110110010 1100101001101
= 434 =6577
Similarly, the other pairs can also be generated in the following way. Now after generating all the numbers by applying crossover and mutation on each the pair we get; 434, 6577, 263,5798,8069,9202,4478,816,3646,2605
After the second generation we continue with the 3rd , 4th and 5th generation to generate 50 numbers (Each generation 10 populations) and get the final set of numbers

Step-2 Encryption:1. Once all the numbers are generated then let this array of numbers be called SUB_ARRAY and select the first digit of each number from SUB_ARRAY and a new collection of numbers is generated and let this collection is called
COLLECTION_ARRAY.
2. Use this numbers from COLLECTION_ARRAY sequentially for substituting on a one-to-one basis for the characters of the plain text.
3. Use ASCII values of the plain text characters and subtract the numbers of COLLECTION_ARRAY from the ASCII values.
For example the message "SOUMYA" the CIPHER TEXT will be calculated according to following method.
LET SUB_ARRAY = {4167, 10117, 5602, 4867, 4307, 2452}

ENCRYPTION:

Character ASCII Value
Collection_Array
Number taken sequentially Subtract Result
S-> 83 4 83 - 4 79 O ->79 1 79 - 1 78
U ->85 5 85 - 5 80 M ->77 4 77 - 4 73
Y ->89 4 89 - 4 85 A ->65 2 65 - 2 63
The enciphered message is "RESULT"
The Cipher text is: {79, 78, 80, 73, 85, and 63}
Step-3 Decryption: In this section the implementation of different types of files are presented. The text, executable and dynamic link libraries files are taken for experiments. This implementation has been done using high-level language.

79 83 -S	78 79 -O	80 85- U
73 77 -M	85 89 -Y	63 65 –A

G.Trapdoor Generation:

In MKQE, when a multi-keyword query q comes, a query vector is made utilizing the method talked about as a part of the past segment. Once more, v quantities of these dummy locations are situated to 1 and all the remaining locations are situated to 0. We encrypt the query vector utilizing the Matrix-Storage algorithm as described previously. The query vector and the inverse of a matrix are utilized as two parameters of the algorithm. MKQE creates a score to focus the location of a file in the coordinating result set. For a document with a index pi, its score is calculated as follows.
Pi ●q= r(Xi+∑ €i(v)) + ti

H.Query:

The DC sends the trapdoor T and parts of his attributes to the CSP. With the data, the CSP firstly figures out which files can be gotten to by the DC, and after that processes the matching score of each authorized file in the encrypted index set I. After that, the CSP sorts the outcomes in based of the scores, and just returns the top k files in the subsequent set to the DC. In our trapdoor algorithm, the score is calculated utilizing the Formula .When the keyword weight is considered as, the estimations of the locations in the query vector are likely determined by their relating weights. In this situation, the scores is computed using the Formula.

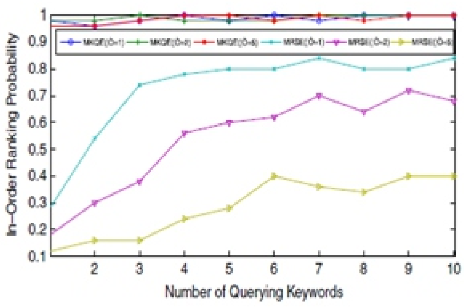

Fig. 4. The in-order ranking probability with different σ

I.In-order result evaluation:

We likewise think about the in-order ranking performance in the returned result set of MKQE with MRSE. In this experiment, we set the quantity of file indexes is 1000, the quantity of keywords in the dictionary is 1000, and the quantity of keywords in a query varies from 1 to 10. The sum of the dummy keywords takes after the ordinary distribution with the mean u = 0. We pick three distinctive variance σ: 1, 2 and 5. For every query, the main 50 ranked files are returned. As we said above, for two files i and j, if Ii matches less keywords and gets a lower score than Ij, we call it is an in order ranking result. Something else, an out-of-order error happens. Fig. 4 shows the outcome. As we can see from the outcomes, for MRSE, the bigger the variance σ, the bring down the probability of the in-order ranking performance Moreover, when the quantity of keywords in the query gets to be littler, the probability turns to be even lower. . For case, when there is just 1 keyword and the variance is 5, the in-order ranking probability is under 5% in MRSE.

Not with standing for the queries with 10 keywords, the in-order ranking probabilities are just 82%, 60% and 40% for the difference 1, 2 and 5, individually. Such an outcome is not exceptionally acceptable; the DC has a really high opportunity to miss the files they truly needs in the top k areas. While in MKQE algorithm, regardless of what number of keywords are incorporated in a query, and regardless of what is the change σ, the in-order probability is dependably over 95%. Particularly, when the quantity of keywords is large (≥6), MKQE can effectively distinguish all the coordinating files with the in-order probability near 100%. In another word, the DC has extremely high probability to retrieve every one of the files it truly needs.

J.Keyword Access Frequency Analysis:

Numerous examination works have concentrated on the multi-keyword query issue and discovered that data recovery applications regularly have power-law constraints (otherwise called Zipf's Law and long tails), and the access frequency (weight) dependably takes after the overwhelming tail distributions. Ordinarily, a DC likes to see the files which have high weight keywords. MKQE consider this variable when generating the query result. In this arrangement of trials, we compare the query results in MRSE and MKQE when taking keyword weights into accounts. 1000queries are executed, and each query has 4 keywords.

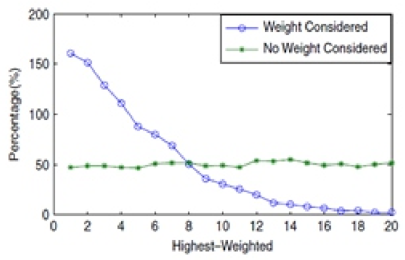

Fig. 5. Percentage of files containing highest-weighted keywords in the top 10 locations

Fig. 5 demonstrates the outcome. X axis shows to the keywords requested by their weights. The first has the most elevated weight by and large. Y axis demonstrates the rate of the returned records in which those the keywords shows up. As should be obvious from the figure, in MRSE, regardless of how popular a keyword is, just 5% of the returned files have one of these keywords. While in MKQE, a keyword with a bigger weight has much higher probability to show up in the outcome set.

V.CONCLUSION:

In this paper, we expect to give a reasonable answer for multi keyword ranked query problems over encrypted data in the cloud environment. We first characterize the issue, break down the current solutions, and design a novel algorithm called MKQE to address the issues. MKQE utilizes an partitioned matrices approach. At the point when the measure of encrypted data increments and more keywords are should be presented, the searching framework can be normally extended with the minimal overhead. We additionally outline another trapdoor generation algorithm, which can take care of the out-of-order" issue in the returned result set without losing the information security and protection property. Besides, the weights of the keywords are taken over in the ranking algorithm when producing the query result. The DC has high probability to recover the records they truly require. The simulation experiments confirm that our methodology can accomplish better execution with an agreeable security level.In the future, we will explore new approaches to the further enhance multi-keyword query capabilities. By using high security algorithms'.

REFERENCES:

[1] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval.J.ACM, 45:965{981, November 1998.

[2]C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. In ICDCS'10, pages 253{262, 2010.

[3] S.Kamara and K. Lauter,"Cryptographic Cloud Storage ," Proc. 14th Int'l Conf. Financial Cryptography and Data Security, Jan. 2010.

[4] D. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," Proc. IEEE Symp. Security and Privacy, 2000.

[5]E.-J. Goh, "Secure Indexes," Cryptology ePrint,Archive,http://eprint.iacr.org/2003/216. 2003.

[6] Y.-C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," Proc. Third Int'l Conf. Applied Cryptography and Network Security, 2005.

[7] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public Key Encryption with Keyword Search," Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2004.

[8] 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 '10), 2010.

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

[10] L. Ballard, S. Kamara, and F. Monrose, "Achieving Efficient Conjunctive Keyword Searches over Encrypted Data," Proc. Seventh Int'l Conf. Information and Comm. Security (ICICS '05), 2005.

[11] D. Boneh and B. Waters, "Conjunctive, Subset, and Range Queries on Encrypted Data," Proc. Fourth Conf. Theory Cryptography (TCC), pp. 535-554, 2007.

[12] E. Shen, E. Shi, and B. Waters, "Predicate Privacy in Encryption Systems," Proc. Sixth Theory of Cryptography Conf. Theory of Cryptography (TCC), 2009.

[13] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: IEEE Conference on Computer Communications, INFOCOM'11, 2011, pp. 829–837.

[14] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, in: Information Processing and Management, 1988, pp. 513–523.

[15]J. Zobel, A. Moffat, Exploring the similarity space, Sigir Forum 32 (1998) 18–34.

AUTHORS:

SHAIK.ASIFA has received her B.Tech degree in Information Technology from Jawaharlal Nehru Technological University, Anantapur, and is pursuing M.Tech in Computer Science & Engineering from the Audisankara College of Engineering and Technology (Autonomous),Gudur, Affiliated to the Jawaharlal Nehru technological university, Anantapur.

Prof. Rajendra Chadalawada has received his bachelor's degree and MCA from Sri Venkateswara University, Tirupati. He received M.E (CSE) from Sathyabama University Chennai. He has received one more master's degree M.Tech (CS) from Jawaharlal Nehru Technological University, Anantapur, AP, INDIA. He is currently working as professor and head, Dept of CSE, Audisankara College of Engineering and Technology, Gudur, AP, INDIA.

Dr A. Sri Lakshmi is currently working as lecturer in computer application, Govt. Degree College (w), Srikalahasti, Chittoor Dt. AP India. She has received her bachelor's degree from Sri Venkateswara University, Tirupati,. MCA FROM REC WARNGAL, and M.Tech from Acharya Nagarjuna University Guntur. She was awarded PhD. in computer science from Sri Padmavathi Viswavidyalayam, Tirupati. Her research interests are data mining and cloud computing.