International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 2, Issue 4, April 2015

Ranked Search over Encrypted Cloud Data using Multiple Keywords

[1]Nita Elizabeth Samuel,[2]Revathi B. R,[3]Sangeetha .M,[4]SreelekshmySelvin,[5]Dileep.V.K
[1][2][3][4]LBS Institute of Technology for Women, Trivandrum
[5]Associate Professor,LBS Institute of Technology for Women,Trivandrum
[1]nitalizsam@gmail.com,[2]revbabu74@gmail.com,[3]san.sangeetham@gmail.com,[4]sreelekshmyselvin@gmail.com,
[5]dileepvk@gmail.com

Abstract: **With the advancement in technology, users have the flexibility for outsourcing data from local sites to commercial public clouds. As data is being outsourced to clouds, there is the need for ensuring the security of the data which is being stored in the cloud. We have been using various security measures in the cloud infrastructure for this purpose and that includes encryption of the cloud data and accessing them using a Boolean keyword search or a single keyword. But these two methods have its own drawbacks. As the number of cloud service users increases these two methods of accessing the data will become less secure. So we are introducing the new idea of multi keyword ranked search on encrypted cloud data. In this we implement a set of strict privacy policies for secure cloud data utilization. Coordinate matching which is an efficient method for finding the most relevant data with the help of queries is used here. Similarities between data and the multi keyword being used are found out by using inner product similarity**.

Keywords: **cloud computing, searchable encryption, privacy preserving, multi keyword search, ranked search.**

I. INTRODUCTION

Cloud computing is a promising technology. It provides the user with various types of services like infrastructure as a service (IaaS), software as a service (SaaS) and platform as a service (PaaS).Cloud computing means storing and accessing of data and programs over the internet instead of storing it in your computer's hard drive. Its use is location independent, on demand service used in the "pay as you go" model. Cloud storage means storage of data online in the cloud where the data is stored in and accessible from multiple distributed and connected resources that comprise the cloud. Benefits include greater accessibility and reliability, rapid deployment, data backup, archival and disaster recovery, lowest cost as a result of not having to purchase and maintain expensive personal hardware. However there are potential security concerns. To protect the privacy of data and unauthorized access of data, sensitive data must be encrypted. This encrypted data is outsourced to cloud for storage [1]. To retrieve this stored data searching is done on the encrypted data using keyword. Owing to the large amount of data in the cloud, for efficient retrieval of the required documents and usage of bandwidth multi keyword ranked search can be used.

II. LITERATURE SURVEY

Documents in cloud are searched using keywords and the searching and retrieval must be in a secure manner. Besides the keyword used must also be protected, since it can give much information about the searched documents. The keyword should retrieve the most efficient and relevant data only. Various techniques for searching over encrypted data have been mentioned in the references.

Traditional single keyword searchable encryption in the symmetric key setting was first studied in [2]. It builds an encrypted searchable index such that the contents remain hidden to the servers unless it is given trapdoors via secret keys. Each word in the document is encrypted independently under a special two-layered encryption construction. Different types of searchable encryption were proposed and improvements and advancements were made. Symmetric searchable encryption (SSE) is appropriate in any setting where the party that searches over the data is also the one who generates it. The main advantages of SSE are efficiency and security while the main disadvantage is functionality. The security guarantees provided by SSE are without any tokens the server learns nothing about the data except its length and given a token for a keyword w, the server learns which (encrypted) documents contain w without learning w.

In public key searchable encryption (PSE) [7] sender and receiver are different entities and may not share a secret key. SSE is more efficient than PSE. Further changes were made in [3], proposed to use Bloom filters to construct indexes for the data files. Updates to the index can be done efficiently but search time for the server is slow. To facilitate more efficient search similar 'index' approaches were proposed in [4] and [5]. Here a single encrypted hash table index is used for the entire files. In the index table, each entry would consist of the trapdoor of a keyword and an encrypted set of file identifiers whose corresponding data files contain the keyword. Here also updates to the index are inefficient.

The first searchable encryption scheme in the public key setting was proposed in [6] in light of the advancements made. In this anyone with public key can write to data stored on a server but only authorized users with private key can search the data. This method is expensive and keyword privacy is not protected since server could encrypt any keyword with public key and use the received trapdoor to evaluate the encrypted data. These methods support only the 'exact' keyword search and are not suitable for use in the present scenario. Fuzzy search: A type of search that will find matches even when users misspell words or enter in only partial words for the search proposed in [8]. Keywords are measured using edit distance and fuzzy keyword sets are constructed. Straight forward and wild card based are the two approaches used. Large storage requirements and large overhead are the major disadvantages.

To enrich searching conjunctive keyword search was proposed in [9].If the user is actually interested in documents containing each of multiple keywords (conjunctive keyword search) the user must either give the server capabilities for each of the keywords individually and rely on an intersection calculation (either by the server or the user) to determine the correct set of documents. Predicate encryption schemes were proposed in [10] and [11]. Predicate privacy preserving search on keyword get the better of Public Encryption Keyword System by using randomization technique. In which keywords are randomized and therefore trapdoors does not provide any meaningful keywords. The user and the receiver share a secret key which is not logical when there is huge number of users. It supports both conjunctive and disjunctive search. Conjunctive search returns those documents in which all keywords specified by the search query appears. Disjunctive search on the other hand returns every document that contains a subset of the specified keywords. This method does not support multiple keyword searches and predicate encryption does not perform ranked search.

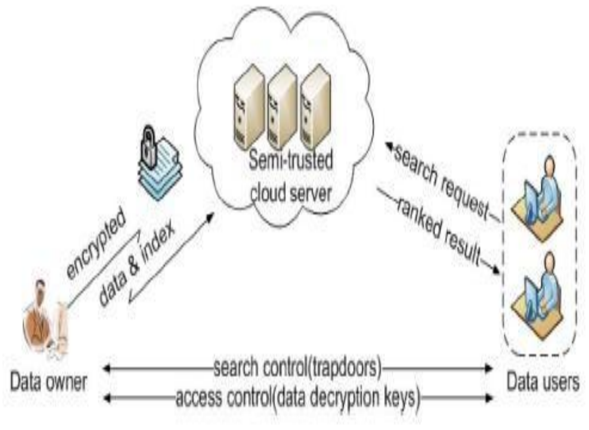
Secure ranked keyword search was proposed in [12]. Data owner has a collection of n data files C = (F1, F2, . . . ,Fn) that he wants to outsource on the cloud server in encrypted form. Before outsourcing, data owner will first build a secure searchable index I from a set of m distinct keywords W = (w1, w2,
...,wm) extracted from the file collection C, and store both the index I and the encrypted file collection C on the cloud server. To search the file collection for a given keyword w, an authorized user generates and submits a search request in a secret form a trapdoor Tw of the keyword w to the cloud server. When the cloud server receives the request from the user it will search the index I and return the corresponding set of files to the user. To reduce bandwidth, the user may send an optional value k along with the trapdoor Tw and cloud server only sends back the top-k most relevant files to the user's interested keyword. Top k retrieval is done as explained in [13].

III. PROPOSED METHOD

To meet the data retrieval need of users, the large amount of documents demand the cloud server to perform result relevance ranking, instead of returning undifferentiated results. Such a ranked search system enables data users to find the most relevant information quickly, rather than sorting through every match in the file collection [14].To improve the search result accuracy and to enhance the user searching experience, it is necessary for such ranking system to support multiple keywords search, as single keyword search often yields far too coarse results. The Coordinate Matching technique can be used for multi keyword search [15].

This paper proposes multi-keyword ranked search over encrypted cloud data. Among various multi- keyword semantics, the efficient principle of "coordinate matching", i.e., as many matches as possible, is used to capture the similarity between search query and data documents, and further use "inner product similarity" to quantitatively formalize such principle for similarity measurement [15].We first propose a basic MRSE scheme using secure inner product computation, and then significantly improve it to meet the privacy requirements of threat model.

Architecture:


Semi-trusted cloud server
encrypted data & index
Search request
ranked result
Data owner
Data users
search control(trapdoors)
access control(data decryption keys)

IV. PROBLEM STATEMENT

A cloud data hosting service involves three different entities, the data owner, the data user, and the cloud server. The data owner has a collection of data documents F to be outsourced to the cloud server in the encrypted form C. To enable the searching capability over C for effective data utilization, the data owner, before outsourcing, will first build an encrypted searchable index I from F, and then 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 user acquires a corresponding trapdoor T through search control mechanisms. Upon receiving T from a data user, the cloud server is responsible to search the index I and return the corresponding set of encrypted documents. To improve the document retrieval accuracy, the search result should be ranked by the cloud server according to some ranking criteria. Also, to reduce the communication cost, the data user may send an optional number k along with the trapdoor T so that the cloud server only sends back top-k relevant documents that is required by the user. Finally, the access control mechanism is employed to manage decryption capabilities given to users and the data collection can be updated in terms of inserting new documents, updating existing documents, and deleting existing documents.

4.1 DESIGN GOALS

Design should simultaneously achieve security and performance guarantees to enable ranked search for effective utilization of outsourced cloud data.

- ➢ Multi-keyword ranked search- To design a search scheme which allow multi-keyword search and provide result similarity ranking for effective data retrieval, instead of returning undifferentiated results.
- ➢ Privacy-preserving- To prevent the cloud server from learning any additional information from the data set and the index, and to meet privacy requirements.
- ➢ Efficiency- Above goals on functionality and privacy should be achieved with low communication and computation overhead.

4.2 THREAT MODEL

The cloud server is considered as “honest-but- curious”. In this threat model, the cloud server is supposed to possess more knowledge than just the encrypted data sets and searchable index. Such information may include the correlation relationship of given search requests (trapdoors), the data set related statistical information. An instance of possible attacks in this case is that the cloud server could use the known trapdoor information combined with document/keyword frequency to deduce/identify certain keywords in the query.

V. MRSE FRAMEWORK

The system model of MRSE is considered to be composed of four algorithms, which are

- Setup: Taking a security parameter ‘l’ as input, the data owner outputs a symmetric key as SK.
- Build Index (*F*, SK): Based on the data set *F*, the data owner builds a searchable index I which is encrypted by the symmetric key SK and then outsourced to the cloud server. After the index construction, the document collection can be independently encrypted and outsourced.
- Trapdoor (W): With t keywords of interest in W as input, this algorithm generates a corresponding trapdoor TW.
- Query (TW,k,I): When the cloud server receives a query request as (TW, k), it performs the ranked search on the index I with the help of trapdoor TW, and finally returns FW, the ranked id list of top-k documents sorted by their similarity with W.

5.1 PRIVACY REQUIREMENTS FOR MRSE

The server should learn nothing about the content being searched by the user [16].

- ➢ Data privacy: The data owner can resort to the traditional symmetric key cryptography (DES/AES) to encrypt the data before outsourcing, and successfully prevent the cloud server from prying into the outsourced data.
- ➢ Index privacy: The searchable index file must also be encrypted because if the cloud server deduces any association between keywords and encrypted documents from index, it may learn the major subject of a document.
- ➢ Keyword Privacy: Users prefer to hide their search content from cloud servers’ i.e. the keywords indicated by the corresponding trapdoor. Though trapdoor can be generated in a cryptographic way to protect the keywords, the cloud server may do some statistical analysis over the search result, for example document frequency is enough to identify keyword with high

probability and this can be used to find the keyword itself.

- Trapdoor Unlinkability: The trapdoor generation function should be a randomized one instead of being a deterministic one. The cloud server should not be able to deduce the relationship of any given trapdoors, for example, determine whether the two trapdoors are formed by the same search request. The deterministic trapdoor generation would give the cloud server advantage to accumulate frequencies of different search requests regarding different keyword(s), which may further violate the aforementioned keyword privacy requirement. Therefore we must introduce sufficient nondeterminacy in trapdoor generation process. This can be done by providing key to each of the users by generating a one-time password (OTP).

VI. COORDINATE MATCHING AND INNER PRODUCT SIMILARITY

For efficient multi keyword ranking we can use the concept of inner product similarity [15], to quantitatively evaluate the efficient similarity measure "coordinate matching. The more terms that appear, the more likely it is that the document is relevant. This approach is called *coordinate matching* [15]. The query becomes a hybrid, intermediate between a conjunctive AND query and a disjunctive OR query.

d	Document D_d
1	Pease porridge hot, pease porridge cold,
2	Pease porridge in the pot,
3	Nine days old.
4	In the pot cold, in the pot hot,
5	Pease porridge, pease porridge,
6	Eat the lot.

Table 1: A small document collection

Consider, for example, the six documents shown in Table 1. For the query *eat*, it is clear that document 6 is the best and only answer. But what about the query *hot porridge?*. In a conjunctive Boolean sense, document 1 is the only answer. But three other documents might also be relevant, and coordinate matching yields a ranking $D1 > D2 = D4 = D5 > D3 = D6 = 0$. Documents containing only one of the terms are available as answers, should the user wish to view them.

This process can be formalized as an inner product of a query vector with a set of document vectors. Table 2a shows the same collection, with a set of binary document vectors represented by *n* components, *n* being the number of distinct terms in the collection. The two example queries can also be represented as n-dimensional vectors and are shown in Table 2b.

The similarity measure of query Q with document Q_i is expressed as

$M(Q, D_d) = Q.\ D_d$

(a)	*d*	Document vectors $\langle w_{d,t} \rangle$									
		col	*day*	*eat*	*hot*	*lot*	*nin*	*old*	*pea*	*por*	*pot*
	1	1	0	0	1	0	0	0	1	1	0
	2	0	0	0	0	0	0	0	1	1	1
	3	0	1	0	0	0	1	1	0	0	0
	4	1	0	0	1	0	0	0	0	0	1
	5	0	0	0	0	0	0	0	1	1	0
	6	0	0	1	0	1	0	0	0	0	0
(b)	*eat*	0	0	1	0	0	0	0	0	0	0
	hot porridge	0	0	0	1	0	0	0	0	1	0

Table 2: Vectors for inner product calculation a) Document vector b) query vector

The *inner product* of two n-vectors $X = <x_i>$ and $Y = <y_i>$ is defined to be

$$X.Y = \Sigma_{i=1}\ x_i.y_i$$

If D_i is the binary data vector for a particular document F_i, then each bit $D_i[j] \epsilon \{0,1\}$ represents the presence or absence of the corresponding keyword W_j in that document.Q is the binary query vector indicating the keywords of search where each bit $Q_j \epsilon \{0,1\}$ represents the existence of the corresponding keyword Wj in the query. The similarity score between the documents F_i to the query is computed as the inner product of binary column vectors D_i.Q. For the purpose of ranking the cloud server compares the similarity of query with different documents. For example,

M(hot porridge, D1) = (0,0,0,1,0,0,0,0,1,0). (1, 0, 0, 1, 0, 0, 0, 1, 1, 0) =2

This is without within document frequency $f_{d,t}$. When that is considered the above example can be written as

M(hot porridge, D1) = (0,0,0,1,0,0,0,0,1,0). (1, 0, 0, 1, 0, 0, 0, 2, 2, 0) = 3

VII. INDEXING

Inverted indexing is used which is also referred to as cross reference [15]. In this the keywords are listed in alphabetical order together with their location where they appear. The intended solution is usually some kind of dynamic dictionary data structure such as a hash table or binary search tree used to record the distinct terms in the collection, with a linked list of nodes storing line numbers associated with each dictionary entry. Once all documents have been processed, the dictionary structure is traversed, and the list of terms and corresponding document numbers is written.

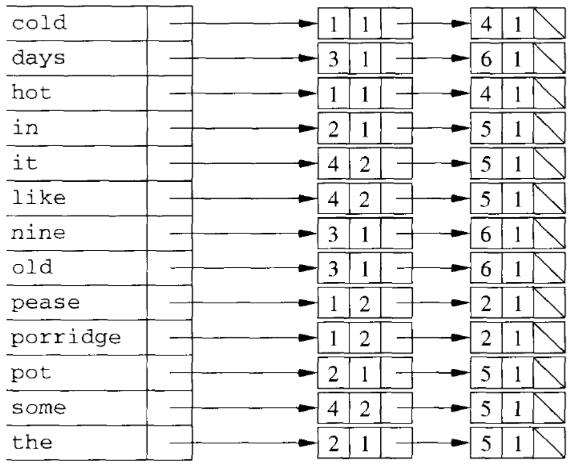


Table 3: Data structure representing inverted file.

The list for a particular keyword can contain details such as:

1. File ids of the files which has the particular keyword
2. Term frequency for each file which denotes the number of times the keyword has occurred in the file. This measures the importance of the keyword in that file.
3. Length of each file
4. Relevance score for each file
5. Number of files that have the particular keyword.

VIII. RANKING

When one keyword appears in most documents in the data set, the importance of this keyword in the query is less than other keywords which appears in fewer documents. Similarly, if one document contains a query keyword in multiple locations, the user may prefer this to the other document which contains the query keyword in only one location. To capture these information in the search process, we use the TF X IDF weighting rule within the vector space model to calculate the similarity, where TF (or term frequency) is the number of times a given term or keyword appears within a file (to measure the importance of the term within the particular file), and IDF (or inverse document frequency) is obtained by dividing the number of files in the whole collection by the number of files containing the term. Score can be calculated as

$$Score(F_i, Q) = \frac{1}{|F_i|}\sum_{W_j \in \widetilde{W}} (1 + \ln f_{i,j}) \cdot \ln\left(1 + \frac{m}{f_j}\right).$$

Here fi,j denotes the TF of keywords Wj in file Fi, fj is the number of files that contain keyword Wj which is the document frequency, m denotes the total number of files in the collection and |Fi| is the Euclidean length of file obtained by

$$\sqrt{\sum_{j=1}^{m} (1 + \ln f_{i,j})^2},$$

which functions as the normalization factor.

If in BuildIndex, for every keyword Wj appearing in the document Fi, the corresponding entry $D_i[j]$in the data vector D_i is changed from a binary value 1 to the normalized term frequency, i.e. $\frac{1+\ln f_{i,j}}{|F_i|}$ similarly, the query vector Q changes corresponding entries from 1 to $\ln(1 + \frac{m}{f_j})$.

Finally, the similarity score is as follows

$I_i \cdot TW=$

$$\begin{aligned} &= r(D_i \cdot Q + \varepsilon_i) + t \\ &= r\left(\sum_{W_j \in Q} \frac{1 + \ln f_{i,j}}{|F_i|} \cdot \ln\left(1 + \frac{m}{f_j}\right) + \varepsilon_i\right) + t \\ &= r(Score(F_i, Q) + \varepsilon_i) + t. \end{aligned}$$

Therefore, the similarity of the document and the query in terms of the cosine of the angle between the document vector and the query vector could be evaluated by computing the inner product of sub index I_i and trapdoor TW.

Word	w_i				
File ID	F_{i_1}	F_{i_2}	F_{i_3}	$\cdots$	$F_{i_{N_i}}$
Relevance Score	6.52	2.29	13.42	4.76	13.80

Table 4: Inverted index with scores

CONCLUSION

Multi keyword ranked search is defined under strict privacy requirements. Among various multi-keyword semantics, we choose the efficient similarity measure of "coordinate matching," i.e., as many matches as possible, to effectively capture the relevance of outsourced documents to the query keywords, and use "inner product similarity" to quantitatively evaluate such similarity measure. The proposed scheme would introduce only low overhead on computation and communication and is effective for a 'pay as you go model'.

REFERENCES

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

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

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

4. 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.

5. R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption Improved Definitions and Efficient Constructions," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), 2006.

6. 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.

7. M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and Efficiently Searchable Encryption," Proc. 27th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '07), 2007.

8. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy Keyword Search Over Encrypted Data in Cloud Computing," Proc. IEEE INFOCOM, Mar. 2010.

9. P. Golle, J. Staddon, and B. Waters, "Secure Conjunctive Keyword Search over Encrypted Data," Proc. Applied Cryptography and Network Security,pp 31-45,2004.

10. J. Katz, A. Sahai and B. Waters, "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products," Proc. 27th Ann Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT), 2008.

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

12. 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.

13. S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+r: Top-k Retrieval from a Confidential Index," Proc. 12th Int'l Conf. Extending Database Technology (EDBT '09), pp. 439-449, 2009 .

14. A. Singhal, "Modern Information Retrieval: A Brief Overview," IEEE Data Eng. Bull., vol. 24, no. 4, pp. 35-43, Mar. 2001.

15. I.H. Witten, A. Moffat, and T.C. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann Publishing, May 1999.

16. N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," Proc.IEEE INFOCOM, pp. 829-837, Apr, 2011.