Fast Information Retrieval over Encrypted Outsourced Cloud Data

Vasudha Arora[1(✉)] and Shweta Mongia[2]

[1] Department of Computer Science and Engineering, GD Goenka University, Gurugram, India
vasudharora6@gmail.com
[2] Department of Informatics, UPES, Dehradun, India

Abstract. The data used in cloud applications is directly exposed to the cloud service provider, and because of the potential compromise of the cloud, could also be learned by adversaries. When encrypted data is hosted on cloud provided that there are large amount of data files, utilization of encrypted data effectively becomes a very challenging task. In a cloud computing environment, where outsourced data of organizations is shared with a large number of users. These variety of users might be interested in retrieving certain specific data files during a given session. A popular and interesting way to do so is by using keyword-based search. These search techniques facilitate users to search and retrieve data files selectively in which the users are interested. These keyword-based searches are being widely used for plaintext searches. But data encryption poses a challenge to perform keyword search using existing paintext search methods to be used for encrypted outsourced data on cloud. In this paper, we have analyzed the searchable indexes that could be used to make a fast and effective search on encrypted outsourced data and proposed a scheme that could make fast and accurate searches over encrypted outsourced cloud data. Simulation results have revealed that the proposed scheme takes much less time in generating the searchable index as compared to already existing techniques. The vector space model being used earlier for keyword based searches on encrypted data, is relatively time consuming and hence leads to very high time complexity during relevance score calculations as well as index generation for large datasets. Hence the proposed scheme achieves a fast and secure relevance scoring for large number of datasets also and in much less time as compared to the vector space model.

Keywords: Data outsourcing · Inverted indexes · Searchable encryption

1 Introduction

Data outsourcing is a service provided by a CSP to store the data of the organizations on the cloud server itself. However, such outsourcing raises some serious issues of securing the privacy of outsourced data.

U. Batra et al. (Eds.): REDSET 2019, CCIS 1230, pp. 100–109, 2020.
https://doi.org/10.1007/978-981-15-5830-6_9

As a new innovation, major IT vendors such as IBM, Microsoft etc. are now incorporating the facility of data outsourcing in their service offerings. Outsourcing is an IT facility provided by cloud technology, where an individual or an organization is able to store their somewhere else using the Internet on a pay per use basis. The term 'cloud computing' as defined by NIST [1] as a "model for enabling convenient, on-demand network access to a shared pool of configurable computing resource…. that can be rapidly provisioned and released with minimal management effort or service provider interaction".

With cloud computing almost every IT facility is provided to its users over the Internet as a service which includes infrastructure including storage & servers, platforms, memory applications etc. Cloud is capable of offering us the private clouds as well as public clouds where we can outsource our data depending on its sensitivity. Outsourcing to a public cloud, where the provider serves multiple customers simultaneously using resource pooling, may share many of the risks of traditional outsourcing. These risks are difficult to alleviate using contract negotiation due to limited opportunities to customize the service delivery.

2 Inverted Index

An inverted index [10] is a data structure that is virtually used in most of IR systems. It is sometimes also known as an inverted file. In a collection C of text files, an inverted index contains the information about mapping of the terms in a file to their corresponding location of occurrence. In contrast to a forward index, that is used to store the information about mapping a particular document to its contents, an inverted index stores a mapping from content to its location. These are generally used for fast full text searches.

When a data user wishes to access the outsourced cloud data using queries, inverted indices are more practical for large collections. For any information retrieval problem one cannot predict the keys in advance that people will use in queries. Therefore, every word in a given document is an equally important search term and the only feasible solution is to index by all keys (words). The big advantage of inverted indexes over forward or normal indexes is that they're excellent for representing values which are appearing frequently and hence a good candidate for search engines.

The big downside of inverted indexes is their fastest implementations are hard to update, and often have to be fully rebuilt every time the database is updated. In practice, most relational databases that implement these types of indexes are columnar databases, which implement the whole table using inverted index structures to store the column values (Fig. 1).

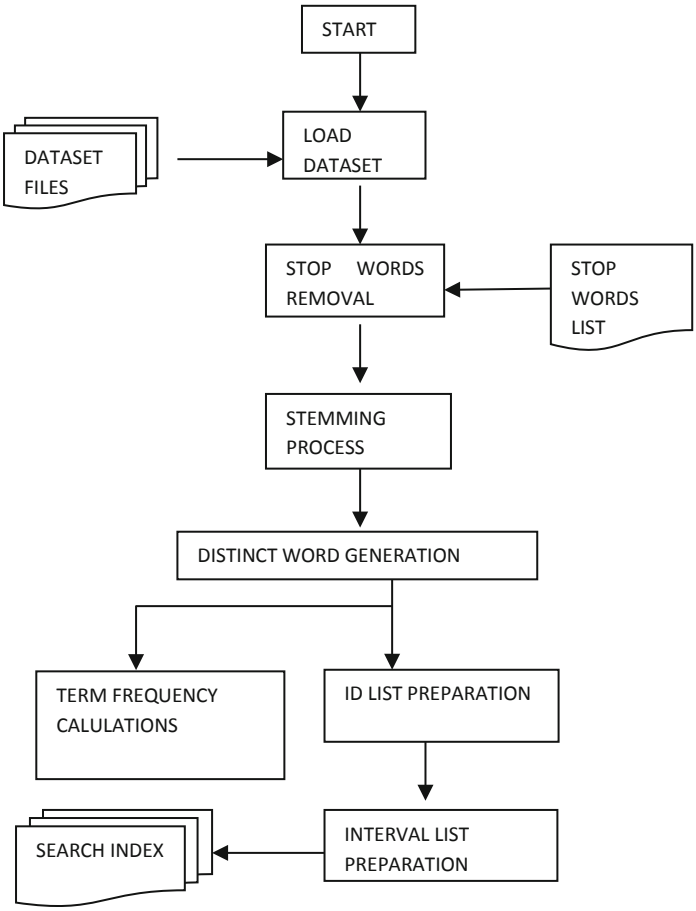

Fig. 1. Process for creating inverted index

3 Information Retrieval

Information retrieval (IR) could be stated as an activity of finding documents or contents which is unstructured (usually text) [10]. Search is a central part in information retrieval whose goal is to predict the relevant documents as per needed by the user. An Information retrieval model must encompass documents, queries by user and if possible some sort of ranking functions to rank the documents retrieved according to relevance for users.

Information retrieval from inverted indexes could be Boolean or statistical. A Boolean retrieval is based on whether the keywords entered in the search query are present or absent in the documents or statistical that applies certain rank order criteria in order to arrange the documents according to their relevance to the user.

4 Vector Space Model

In a vector space everything, such as words, documents, queries and even user preferences, is a vector in some high dimensional space. In order to understand a vector space model one should know what are the dimensions of that space, how to project words, documents and queries to that space and then finally how to compare documents and queries. In a vector space every document represents a new dimension and hence the number of dimensions is constantly growing. Therefore for m documents in the document collection we have an n dimensional vector space. The terms in the documents represent the axes of the space and documents are points or vectors in this space. When this model is applied to a web search engine a very high dimensional vector space is created consisting of tens of millions of dimensions. The vector model created is very sparse that contains a number of zeros.

If we have this vector space of documents we also represent the queries by users as vectors in the space and rank order according to their proximity, to the query in this space, where proximity refers to the similarity of vectors which can be calculated as inverse of Euclidean distance between the two vectors. This is done to come out of the Boolean model and to rank more relevant documents higher than less relevant documents. Using Euclidean distance may always not be a good idea because Euclidean distance is large for vectors of different lengths (Table 1).

Table 1. Term-document incidence matrix (Vector Space Model Information Retrieval)

Documents Terms	Research foundation F1	Network technology F2	Abstract awards F3	Cloud technology F4
Internet	0.79	0.69	0.423	0.004
Spy	0.197	0.78	0	0.645
Teaching	0.231	0	0	0.254
Beware	0.85	0	0.466	0
Domain	1.987	0	0.120	0.342

The Fig. 2 shows the vector space model for two terms Internet and Domain representing two different axes in the vector space. Here, the document vector d1 is closer to the term Network and nothing to do with Domain while d3 is closer to the term Domain and nothing to do with Network. If we want to find out a document that contain both the terms the document d2 should be the answer to the query.

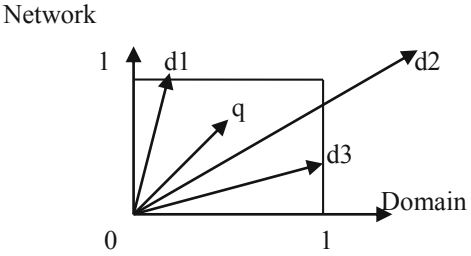

Fig. 2. Vector space model

Using Euclidean distance between q and d2 for calculating relevance of documents based on queried keywords may not be a good choice as the Euclidian distance between them is large even though the distribution of terms in the query q and distribution of terms in the document d2 are very similar. The cosine similarity between the document and the query is calculated. As a ranking function we can use tf-idf weighting scheme.

In order to retrieve information a function for ranking the documents [10] known as tf-idf rule is used. Depending on the number of times a term appears in file or a document, each term is assigned a weight. A score between the query term and the document is calculated.

Let, for any keyword in a query, a keyword t present in a file f has a term frequency denoted by $tf_{t,f}$. In order to calculate the term frequency, we ignore the exact ordering of terms in a file and we count number of times a term appears in it.

df_t the document frequency can be calculated as the total files in a collection of files denoted as C in which the keyword t appears. If in a collection of files, denoted as C, there are m files, an IDF the inverse document frequency of a term t is calculated as

$$\text{idf}_\text{t} = \log m\, df_t$$

Now tf-idf rule is used to assign the weight to a term t in a file f as

$$\text{tf-idf}_\text{t,f} = \text{tf}_\text{t,f} \times \text{idf}_\text{t}$$

The relevance score for file f is calculated as the sum of tf-idf weight for each term in a file. Therefore,

$$\text{Relevance Score}_{\text{f,q}} = tf - idft \quad t\varepsilon q \tag{1}$$

After calculating the scores, top-k documents with highest score are picked up and presented to the user.

5 Related Work

Before being outsourced, data is encrypted by the users. Traditional search algorithms for searching the data based on some keywords fail to search on this encrypted data and also raise a concern for privacy of keywords being searched [2].

In order to search the keywords an inverted index is created [9, 10]. An inverted index is a matrix that contains the list of all the unique terms that appear in any document in the collection, and for each term, a list of the documents in which it appears.

A VSM (Vector Space Model) was used by TRSE scheme [6] where authors represented each file in the collection as a vector and each term appearing in the file a new dimension for the vector. If a term appears in the file authors assigned a non-zero value to it otherwise a value zero is assigned for each term. Similarly, the query generated by user is represented as a vector. For each file if a term that appears in the query a non-zero value is assigned otherwise for a term present in file but not in query a zero value is assigned to it.

6 Proposed Scheme

A scheme based on rank order search is proposed for searching multiple keyword. A $O(n_t \times 3)$ order index is created. The proposed scheme [11] is described as follows: An entity data owner has a file collection containing m number of represented as F = {f1, f2, f3, ….., fm) that are required to be outsourced to the cloud server. The tokenization process is applied to chunk the document into terms or tokens before outsourcing. AES is used for symmetrically encrypting F. After removal of defined stop words from the file tokens are collected into an index table in sorted sequence. The sorting is done so that similar identifiers from different files with different relevance scores must be collected together in the index. This helps in reducing search time. The index table contains n_t rows and 3 columns containing the tokens, file identifiers, relevance score of each file with respect to the token. The tokens are represented as a set of n values T = (t1, t2, t3, …, tn). The proposed scheme creates an inverted index having file id, index term and relevance scores. In contrast to vector space matrix it creates index as shown in figure below.

Whenever a data owner uploads a new file or deletes an existing file on the cloud server the server updates the relevance scores. If there is any change or modification in an already existing file a new copy of file is created with a new file identifier and older one is automatically removed from the database (Table 2).

Table 2. Index Structure for proposed scheme

Term	File Id	Rel. Score
Network	F1	0.996
Network	F2	0.993
Network	F3	0.234
Network	F4	0.034
Species	F1	0.087
Species	F2	0.876
Species	F4	0.456
Technology	F1	0.017
Foundation	F1	0.523
Foundation	F3	0.466
Domain	F1	1.987

7 Experimental Evaluation

We created an experimental evaluation for generating the index using vector space model and our proposed scheme for overall performance evaluation on a real data set: National Science Foundation Research Awards Abstracts 1990–2003 [12]. Our experiment environment includes data owner, data users and a cloud server. We used c#. Net platform on a windows 7 machine with core i5 processor. The doubly encrypted index, I' and the encrypted collection of files F' is stored on the commercial public cloud on a virtual instance hired from Microsoft Azure. The client application was installed on a machine(with windows 7 operating system and core i5 processor) and overall scenario was simulated on c#. Net platform.

We evaluated the performance based on following parameters:

a) **Time to Generate Index**

Time taken to create the inverted index and to calculate respective relevance scores is taken into consideration here.

In a vector space model, a term document incidence matrix is created where each term represents a new dimension to the document vector. Hence, the complexity of creating such a vector is of the order O (D $\times$ T). Where D represents the number of document to be outsourced and T represents the number of tokens to be uploaded in the index. For 5 documents having 500 tokens vector space model requires 2500 elements to be uploaded. Whereas, for the inverted index created for the proposed scheme the complexity is of the order O (nt $\times$ 3), where nt represents the number of rows in the index for T number of tokens and there are 3 columns in the generated index. For 5 documents having 500 tokens, the number of terms (representing number of rows) in index is 643, the proposed scheme needs to upload 1929 elements (Fig. 3).

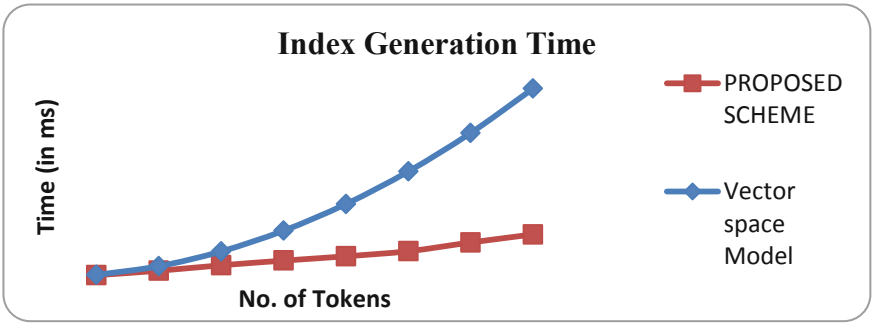

Fig. 3. Time taken to generate index on the scale of number of tokens.

For 5 documents having 500 tokens vector space model requires 2500 elements to be uploaded it takes 142 ms to generate the index. Whereas, for the inverted index created for the proposed scheme the complexity is of the order O (nt × 3). For same number of documents having same number of tokens, the proposed scheme needs to upload 1929 elements can be uploaded in 129 ms. Whereas, with D = 1,000 having approximately 20000 tokens D × T i.e. 1,000 × 20000 = 20,00,000 (approx) elements and these elements takes 1852 s for index generation. For the same set of 1000 files, with 20000 distinct tokens, having approximately 300000 terms, C = 3, nt × C = 300000 × 3 = 9,00,000 elements takes 568 s to compute an index generation in proposed scheme. Hence vector space model, index generation time is more in comparison with proposed scheme.

b) **Score Calculation on retrieval (Search Efficiency)**

In the ScoreCalculate stage, for vector space model, dot product of the query vector from the query with each row in the encrypted index I' is calculated by the cloud server (Fig. 4).

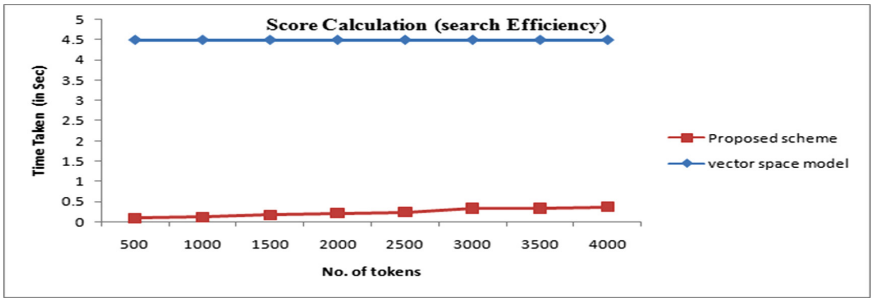

Fig. 4. Time taken to search queried keywords based on the number of tokens.

To calculate the inner product, for an n-dimensional query vector, each row needs n multiplications and n − 1 additions. Hence, the complexity to calculate relevance scores is O(nm) for *m* files in the document collection and n keywords in the collection.

For the proposed scheme, no vector product is being generated and the proposed scheme uses binary search trees for searching the encrypted queried keywords in the encrypted index, hence for j number of terms in the encrypted request generated, the time complexity to search the required keywords and calculate the relevance scores is O(j(log nt)).

8 Conclusions

Rank ordered information retrieval in a secure way according to the relevance of documents to the users of data outsourced to the cloud is one of the major issues in cloud computing environment. A detailed survey of existing IR schemes was done. Retrieving data efficiently from cloud is one of the major issues, which makes the clients reluctant to store their data in cloud environment. Searching over encrypted data has been made possible by using vector space model and inverted indexes. For, VSM, queries and documents are converted to vectors in encrypted form and their dot product of relevance scores is calculated using the cosine similarity algorithm that provides the users with relevant documents. Number of encryption schemes could be applied such as OPE (order preserving encryption), homomorphic encryption etc. to encrypt the data and use the vector space information retrieval model. The proposed scheme creates an inverted index that makes information retrieval possible to the user comparatively faster, simpler in a secure way than using a vector space model for index generation. The proposed scheme is not only secured but has reduced the time complexity and space complexity to much larger extent. In contrast to earlier proposed schemes the proposed one can be applied to huge datasets also. The simulation results show that the vector space model for indexing does not works effectively for huge datasets and take huge time that becomes out of scope for calculations on the application created. The proposed scheme works for text files only. This could be extended to support various file formats including images for improving access control to enhance the security. One of the missing functions in current secure indexes is phrase search. Current multi-keyword search schemes are capable of testing the existence of the query keywords but not able to tell the relative positions of the query keywords. Because of the importance of the phrase search, one of the future research objectives is to provide the capability of phrase searching on data in encrypted form in a secure manner.

References

1. Mell, P., Grance, T.: The NIST definition of cloud computing, National Institute of Standarads and technology, vol. 53, no. 6, pp. 1–50 (2009)
2. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure ranked keyword search over encrypted cloud data. In: Proceedings IEEE 30th International Conference on Distributed Computing Systems, ICDCS 2010, pp. 253–262 (2010)

3. Song, D., Wanger, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings IEEE symposium Security and Privacy, Washington DC, pp. 44–56 (2000)
4. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1467–1479 (2012)
5. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Privacy-preserving multikeyword ranked search over encrypted cloud data. In: Proceedings IEEE INFOCOM (2010)
6. Yu, J., Lu, P., Zhu, Y., Xue, G., Li, M.: Toward secure multikeyword top-k retrieval over encrypted cloud data. IEEE Trans. Dependable Secure Comput. **10**(4), 239–250 (2013)
7. Arora, V., Tyagi, S.S.: Analysis of symmetric searchable encryption over encrypted cloud data. Int. J. Comput. Appl. (0975-8887) **127**(12), 46–51 (2015)
8. Joy, E.C., Indira, K.: Multi keyword ranked search over encrypted cloud data. Int. J. Appl. Eng. Res. **9**(20), 7149–7176 (2014)
9. Singhal, A.: Modern information retrieval: a brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
10. Manning, C.D., Raghavan, P., Schutze, H.: An Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008). ISBN: 0521865719. Online edition©
11. Arora, V., Tyagi, S.S.: An efficient multi-keyword symmetric searchable encryption scheme for secure data outsourcing. Int. J. Comput. Netw. Inf. Secur. **8**(11), 65–71 (2016)
12. NSF Research Awards Abstracts 1990–2003 (2013). http://kdd.ics.uciedu/databases/nsfabs/nsfawards.html
13. Pallickara, S.L., Pallickara, S., Zupanski, M.: Towards efficient data search and subsetting of large-scale atmospheric datasets. Future Gener. Comput. Syst. **28**(1), 112–118 (2012)
14. Raghavendra S., Geeta C.M., Buyya R., Venugopal K.R., Iyengar S.S., Patnaik L. M.: MSIGT: most significant ındex generation technique for cloud environment. In: Proceedings of the 2015 Annual IEEE India Conference, INDICON 2015, Delhi, India, pp 17–20 (2015)
15. Alam, B., Doja, M.N., Mongia, S.: Analysis of security issues for cloud computing. Int. J. Comput. Sci. Inf. Secur. **11**(9), 117–125 (2013)
16. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
17. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_13
18. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1469 (2012)
19. Gupta, B.B.: Analysis of various security issues and challenges in cloud computing environment: a survey. In: Gupta, B., Agrawal, D.P., Yamaguchi, S. (eds.) Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security. IGI Global, Hershey (2016). https://doi.org/10.4018/978-1-5225-0808-3.ch011
20. Ibtihal, M., Driss, E.O., Hassan, N.: Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. Int. J. Cloud Appl. Comput. (IJCAC) **7**(2), 27–40 (2017)