# An Enhanced Keyword Search Using Flexible Ranking Over Differential Query Services in Cloud

**Rasiya V M[1], Shanavas M A[2]**

[1]Computer Science and Engineering, ICET, Muvattupuzha, India

[2]Assistant Professor, Information Technology, ICET, Muvattupuzha, India

**Abstract:** ***Cloud computing has emerged as a major driver for reducing the information technology costs incurred by organizations. In a cost-sensitive environment, an organization is willing tolerate a certain threshold of delay while retrieving information from the cloud. We first review a private keyword-based file retrieval scheme that was EIRQ, users can retrieve the desired percentage of files by assigning ranks to queries. This feature is useful if the user is only interested in a subset of all the matched files .By further reducing the communication cost incurred on the cloud, the existing EIRQ schemes make the private searching technique more applicable to a cost-efficient cloud environment. The main drawback is, we simply determine the rank of each file by the highest rank of queries it matches. Since this ranking is not flexible. In this paper we present a more sophisticated ranking system by providing variable weights to the relevance attributes of each file. A flexible ranked search sorts the matches by relevance. This ranking allows the user to find the most (or least) relevant information quickly, rather than sorting through every match in the content collection. This approach improves the user friendly environment as well as it tries to focus on the reduction of communication cost***

**Keywords:** Cloud computing, Flexible ranking, Rank, Aggregation and distribution layer

## 1. Introduction

Cloud computing as an emanate technology to imperative information technology process in future. Many organizations choose to out-source their data for sharing in cloud. An organization supports the cloud services and authorizes its staff to share files in the cloud, its typical in cloud application. Each file is related by set of keywords. The staff as authorized users is for retrieving files. They can retrieve files of their interests by querying the cloud with certain keywords. Here the key problem is that user privacy. The user privacy is a third party outside the security boundary. The User privacy is classified into two types. 1) Search privacy 2) Access privacy. The cloud knows nothing about what the user is searching for is called Search privacy, and the cloud knows nothing about which files are returned to the user is called access privacy.

Private searching was proposed by Ostrovsky[1] which allows a user to retrieve files of interest from an untrusted server without leaking any information. However, the Ostrovsky scheme has a high computational cost, since it requires the cloud to process the query (perform homo-morphic encryption) on every file in a collection.

To make private searching applicable in a cloud environment, previous work [2] designed a cooperate private searching protocol (COPS), where a proxy server, called the aggregation and distribution layer (ADL), is introduced between the users and the cloud. The ADL deployed inside an organization has two main functionalities: aggregating user queries and distributing search results. Under the ADL, the computation cost incurred on the cloud can be largely reduced, since the cloud only needs to execute a combined query once, no matter how many users are executing queries. Furthermore, the communication cost incurred on the cloud will also be reduced, since files shared by the users need to be returned only once. Most importantly, by using a series of secure functions, COPS can protect user privacy from the ADL, the cloud, and other users.

Differential query services, to COPS, where the users are allowed to personally decide how many matched files will be returned. This is motivated by the fact that under certain cases, there are a lot of files matching a user's query, but the user is interested in only a certain percentage of matched files. This scheme, termed Efficient Information retrieval for Ranked Query (EIRQ), in which each user can choose the rank of his query to determine the percentage of matched files to be returned. The basic idea of EIRQ is to construct a privacy-preserving mask matrix that allows the cloud to filter out a certain percentage of matched files before returning to the ADL. This is not a trivial work, since the cloud needs to correctly filter out files according to the rank of queries without knowing anything about user privacy.

The Three EIRQ schemes based on an ADL are to provide differential query services while protecting user privacy. By using these schemes, a user can retrieve different percentages of matched files by specifying queries of different ranks. By further reducing the communication cost incurred on the cloud, the EIRQ schemes make the private searching technique more applicable to a cost-efficient cloud environment. However, in the EIRQ schemes, we simply determine the rank of each file by the highest rank of queries it matches. In this scheme does not consider the file content or its importance.

In this paper we design a flexible ranking mechanism for the EIRQ schemes to address the above issues. Currently, the ranking of the files on the cloud is determined by the highest rank of the queries matching that file. Since this ranking is not flexible, a more sophisticated ranking system could be developed by providing variable weights to the relevance attributes of each file. This flexible ranked search sorts the matches by relevance. The ranking appears in the Search Results List pane. This ranking allows the user to find the most (or least) relevant information quickly, rather than sorting through every match in the content collection. When you perform a ranked search, you must specify the maximum number of ranked documents (or partitions) you wish to see. This allows to create a top ten list (or a top one hundred list) of the most relevant information pertaining to our search.

This paper is organized as follows. Section II deals with the System model. Section III handles the Scheme description, section IV discusses the proposed method for flexible ranking. Finally in section V concluding remarks are given.

## 2. System Model

The system model consists of three entities. They are Aggregation and Distribution layer, the cloud and the many users. Figure 1 shows that the only one ADL in this paper.

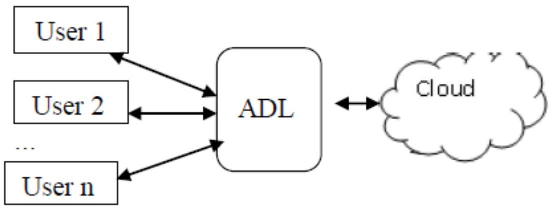

**Figure 1:** System Model

The queries are sending to the ADL by the authorized users. The ADL aggregate users queries and send as combined query to the cloud. Then, the combined queries are processed by the cloud on the file collection and send a buffer. The buffer involve of all matched files to the ADL. The ADL will distribute the search results to each user. In this method the organization may require the ADL to wait for a period of time before running our schemes, which may get a certain querying delay.

## 3. Scheme Description

In this section, the EIRQ scheme described in three schemes.1) EIRQ Efficient,2) EIRQ Simple and 3) EIRQ privacy scheme .By comparing all the scheme the EIRQ Efficient scheme provide less communication cost.

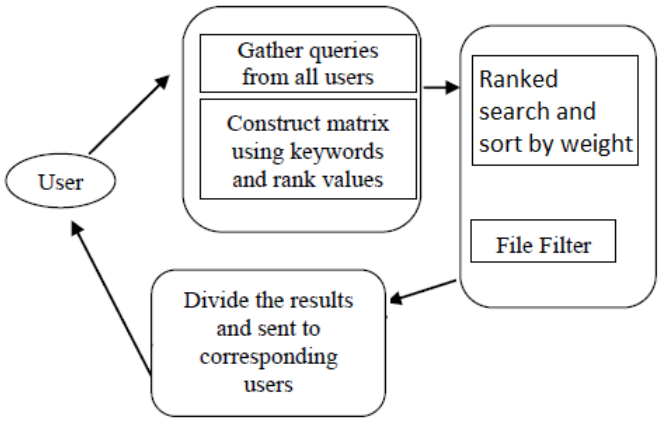
Gather queries from all users
Construct matrix using keywords and rank values
Ranked search and sort by weight
File Filter
User
Divide the results and sent to corresponding users

**Figure 2:** Scheme Description

### 3.1 The EIRQ-Efficient Scheme

The EIRQ-Efficient scheme should be resolved two fundamental problems. First, we should determine the relationship between query rank and the percentage of matched files to be returned. Else those queries are classified into 0 to r ranks. Rank-0 queries have the highest rank and the Rank-r queries have the lowest rank. This relationship by allowing Rank-i queries to retrieve (1-i/r) percent of matched files. Finally Rank-0 queries can retrieve 100 percent of matched files, and Rank-r queries cannot retrieve any files. Secondly, we should determine which matched files will be returned and which will not. In this paper, we simply fix the probability of a file being produces by the highest rank of queries matching this file. Specifically, we first rank each keyword by the highest rank of queries selecting it, and then rank each file by the highest rank of its keywords. If the file rank is i, then the possibility of being filtered out is i/r. Therefore, Rank-0 files will be mapped into a buffer with probability 1, and Rank-r files will not be mapped at all. Since unneeded files have been filtered out before mapping, the mapped files should survive in the buffer with probability 1. EIRQ-Efficient mainly consists of four algorithms. The algorithms are 1) QueryGen 2) Matrix Construct 3)File filter and 4) ResultDivide are easily under-stood. EIRQ-Efficient mainly consists of four algorithms. The algorithms are 1) QueryGen 2) Matrix Construct 3)File filter and 4) ResultDivide are easily under-stood.

Step 1: The user sends the keyword and the rank of the query to the ADL by using QueryGen algorithm.

Step 2: The ADL runs the MatrixConstruct algorithm after aggregating enough user queries, to send a mask matrix to the cloud. The mask matrix M consists that d-row and r-column matrix, where d is the number of keywords, and r is the lowest query rank.

Step 3: The cloud runs the FileFilter algorithm to return a buffer. The buffer contains a certain percentage of matched files to the ADL. Here the DES algorithm used.

Step 4:To distribute search results to each user by the ADL runs the Result Divide algorithm. We require the cloud to attach keywords to the file content to allow the ADL to distribute files correctly. By executing keyword searches the ADL can find out all of the files that match users queries.

## 4. Proposed Model

In the existing work, the EIRQ scheme is proposed to provide a differential query services with the user privacy. It works based on the ranking of users query. In this method the communication cost is also reduced by retrieving only the required contents to the users based on users ranking. Based on this ranking the files will be retrieved to the users. However in this method the ranking of file is done based on only the highest rank of queries it matches. The efficient ranking mechanism has to be implemented in order retrieve the most suitable files to the user.

In our method a flexible ranking mechanisms is used. The more sophisticated ranking system could be developed by providing variable weights to the relevance attributes of each file. A ranked search sorts the matches by relevance. The ranking appears in the Search Results List pane. This ranking allows the user to find the most (or least) relevant information quickly, rather than sorting through every match in the content collection.

### 4.1 Determining Relevance

NXT 4 uses four methods for determining relevance:

**Local frequency**, which means that the more often a query term appears in a document, the more relevant that document is. For example, a document containing five instances of a query term is more relevant than a document containing only two instances of that query term.

**Inverse document frequency**, which means that terms that are rare within the context of the entire site are given a higher relevancy ranking. For example, assume you are searching for two terms, "government" and "policy" within a site containing 20,000 records. If "government" is found only five times and "policy" is found 300 times, then documents containing "government" are given a higher weight when determining relevancy.

**Document length comparison (Density)**, which means that shorter documents (or partitions) containing an equal number of query terms are more relevant than longer documents. Since the terms appear closer together in a shorter document, there is a higher chance that the document is relevant to your search.

**Completeness**, which means that documents containing more of the query terms are more relevant. For example, if your query contains four terms, a document which contains all four terms is more relevant than a record which only contains three terms

### 4.2 Setting the Domain of a Ranked Query

For the typical user, there are two primary uses of domains in ranked queries:

- Ensuring that a particular term or set of terms are included in all search results found by a ranked query.
- Limiting the scope of the query to a subset of the content collection.

To set a domain, we specify a standard query (non-ranked), using any of the standard operators, wildcards, and scopes. All of the documents found by this query are gathered into a single set. The ranked query is then applied to the set. In practice, we can specify a set of terms in the domain that must appear in any search match. For example, if you are searching a content collection of space history, you might set the domain to include apollo and then do a ranked search for space flight missions and moon. All ranked search results are then forced to include the term apollo.

### 4.3 Setting the Weight of an Item in a Ranked Query

Ranked queries look at several different criteria to determine what information is relevant and what information is more relevant than other information. If necessary, you can affect the outcome of a ranked search by giving a particular term or field a higher weight (or multiplier) documents containing weighted terms are given a correspondingly higher score and higher relevance value. We might use weighting when searching a library card catalog to give the Title field of a search a higher relevance value than the Subject field.

To set the weight for a term, you must type the following syntax as part of a ranked query: The default weight is 1; a weight of 2 indicates that the item is twice as important as other items; the maximum weight is 9999). <items to rank> may be one or more of the following:

- term (including wildcards)
- proximity (including phrases)
- field (must specify at least one term in the field)

For example, [Rank 10] [Weight 2: relevance ] rank query indicates that documents containing only the term relevance will be given a higher relative score than those containing only rank or query. The final rank is a composite of several factors, including the frequency of a term in a document and the total number of specified query terms found in the document. In another example, [Rank 15] [Weight 3: [Field Murder Weapons: gun | pistol]] violent crime, the documents containing either gun or pistol in the Murder Weapons field are given a higher relative weight than other items in the query.

Since queries are classified into 0 to 4 ranks, queries in Rank-0, Rank-1, Rank-2, Rank-3, and Rank-4 should retrieve 100 percent, 75 percent, 50 percent, 25 percent,0 percent of matched files, respectively. Let us assume that Alice wants to retrieve 50 percentage of the files (Rank 2) that contain keywords „„A, B’’. The cloud holds 1,000 files, where F1 ,F2 are described by keywords „„A, B’’ and F3,F4 is described by “A,C”,F5 is described by A. It returns F1 and F2 firstly in the search list .ie) `1it is according to the relevance of attribute of file such as if your query contains four terms, a document which contains all four terms is more relevant than a record which only contains three terms.The rank determines the percentages of files retrived in EIRQ.

## 5. Conclusion

In this paper, we proposed a flexible ranking mechanism on EIRQ schemes. By using our schemes, a user can retrieve different percentages of most relevant matched files by specifying queries of different ranks. A ranked search sorts the matches by relevance. This ranking allows the user to find the most (or least) relevant information quickly, rather than sorting through every match in the content collection. By further reducing the communication cost incurred on the cloud, the EIRQ schemes make the private searching technique more applicable to a cost-efficient cloud environment.

## 6. Acknowledgment

The authors would like to thank ICET college authority and the guide for their valuable support. They would also like to thank the reviewers for their valuable feedback and suggestions

## References

[1] Briand, L. C., Daly, J., and Wüst, J., "A unified framework for coupling measurement in objectoriented systems", *IEEE Transactions on Software Engineering*, 25, 1, January 1999, pp. 91-121.
[2] R . Ostrovsky and W. Skeith, "Private searching on streaming data", *Journal of Cryptology*, Volume 20:4, pp. 397-430, October 2007.
[3] Q.Liu,C.Tan,J.Wu,andG.Wang"Cooperative private searching in clouds",*JournalofParallelandDistributedComputing*,2012.
[4] P . P a i l l i e r , "Public-key cryptosystems based on composite degree residuosity classes," *in P r o c . o f E U R O C R Y P T* , 2 0 0 9 .
[5] Q. Liu, C. C. Tan, J. Wu, G. Wang, "Towards Differential Query Services in Cost-Efficient Clouds", *IEEE Transactions on Parallel and Distributed Systems*, Vol. PP No. 99, Year 2013.
[6] Q. Liu, C. C. Tan, J. Wu, G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments", *Proc. of IEEE INFOCOM*, 2012.

## Author Profile

**Rasiya V M** received the Bachelor of Technology degree in Computer Science and Engineering from Mahatma Gandhi University, Kerala. She is currently doing Master of Technology degree in Computer Science and Engineering with Specialization in Information Systems from Mahatma Gandhi University, Kerala.

**Shanavas M A He** is currently assistant professor at ICET, Muvattupuzha, Mahatma Gandhi University, Kerala.