JOURNAL OF INTERNATIONAL ACADEMIC RESEARCH FOR MULTIDISCIPLINARY
Impact Factor 1.393, ISSN: 2320-5083, Volume 2, Issue 1, February 2014

RETRIEVE THE CLOUD DATA IN EFFICIENT WAY USING QUERY SERVICES

DR.B.G.GEETHA*
M.GOVINDARAJ**

*Head of the Department (Academic), K S Rangasamy College of Technology, Tiruchengode, Tamil Nadu, India
**PG Scholar of Computer Science, K.S. Rangasamy College of Technology, Tiruchengode, Tamil Nadu, India

ABSTRACT

Cloud computing as an emerging technology trend is expected to reshape the advances in information technology. In a cost efficient cloud environment, a user can tolerate a certain degree of delay while retrieving information from the cloud to reduce costs. In this the two fundamental issues in such an environment: privacy and efficiency. The first review about the private keyword based file. This scheme allows a user to retrieve files of interest from an untrusted server without leaking any information. The main drawback is that it will cause a heavy querying overhead incurred on the cloud, and thus goes against the original intention of cost efficiency. The new scheme, termed efficient information retrieval for ranked query (EIRQ), based on an aggregation and distribution layer (ADL), to reduce querying overhead incurred on the cloud. In EIRQ, queries are classified into multiple ranks, where a higher ranked query can retrieve a higher percentage of matched files. A user can retrieve files on demand by choosing queries of different ranks. This feature is useful when there are a large number of matched files, but the user only needs a small subset of them. Under different parameter settings, extensive evaluations have been conducted on both analytical models and on a real cloud environment, in order to examine the effectiveness of our schemes.

1. INTRODUCTION

Compute clouds are commonly used by many different users that rely on the existing computing infrastructure to deploy their workloads. Due to the overwhelming merits of cloud computing, e.g., cost-effectiveness, flexibility and scalability, more and more organizations choose to outsource their data for sharing in the cloud. As a typical cloud application, an organization subscribes the cloud services and authorizes its staff to share files in the cloud. Each file is described by a set of keywords, and the staff, as authorized users, can retrieve files of their interests by querying the cloud with certain keywords. In such an environment, how to protect user privacy from the cloud, which is a third party outside the security boundary of the organization, becomes a key problem.

User privacy can be classified into search privacy and access privacy. Search privacy means that the cloud knows nothing about what the user is searching for, and access privacy

means that the cloud knows nothing about which files are returned to the user. When the files are stored in the clear forms, a naive solution to protect user privacy is for the user to request all of the files from the cloud; this way, the cloud cannot know which files the user is really interested in. Private searching means which allows a user to retrieve files of interest from an untrusted server without leaking any information. It requires the cloud to process the query (perform homomorphic encryption) on every file in a collection. It will quickly become a performance bottleneck when the cloud needs to process thousands of queries over a collection of hundreds of thousands of files. Commercial clouds follow a pay-as-you-go model, where the customer is billed for different operations such as bandwidth, CPU time, and so on.

Private searching applicable in a cloud environment, 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.

Introduce a novel concept, 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. let us assume that Alice wants to retrieve 2% of the files that contain keywords "A, B", and Bob wants to retrieve 20% of the files that contain keywords "A, C". The cloud holds 1,000 files, where {F1 , . . . , F500} and {F501, . . . , F1000} are described by keywords "A, B" and "A, C", respectively. In the COPS scheme, the cloud will have to return 1, 000 files. In our scheme, the cloud only needs to return 200 files.

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. Two extensions: the first extension emphasizes simplicity by requiring the least amount of modifications from the Ostrovsky scheme, and the second extension emphasizes privacy by leaking the least amount of information to the cloud. Our key contributions are as follows:

1) The EIRQ schemes based on the ADL to provide a cost-efficient solution for private searching in cloud computing.

2) The EIRQ schemes can protect user privacy while providing a differential query service that allows each user to retrieve matched files on demand.

3) It provide two solutions to adjust related parameters; one is based on the Ostrovsky scheme, and the other is based on Bloom filters.

4) Extensive experiments were performed using a combination of simulations and real cloud deployments to validate our schemes.

II. RELATED WORK

To provide differential query services while protecting user privacy from the cloud. Existing research that is similar to ours can be found in the areas of private searching Unlike searchable encryption, where the user conducts searches on encrypted data, private searching performs keyword-based searches on unencrypted data. Private searching, which allows a server to filter streaming data without compromising user privacy. Their solution requires the server to return a buffer of size O (f log(f)) when f files match a user's query. Each file is associated with a survival rate, which denotes the probability of this file being successfully recovered by the user. Based on the Paillier cryptosystem, the files that mismatch a query will not survive in the buffer, but the matched files enjoy a high survival rate.

Further reduced the communication cost from O(f log(f)) to O(f) by solving a set of linear equations to recover f matched files. However, their scheme requires the decryption of one more buffer, thus the computation cost is higher than the Ostrovsky scheme. Presented an efficient decoding mechanism which allows the recovery of files that collide in a buffer position. Extraction mechanism, which requires a buffer of size O(f) when f files match a user's query. Proposed two new communication-optimal constructions; one uses Reed-Solomon codes and allows for a zero error, and the other is based on irregular LDPC codes and allows for lower computation cost at the server. The above private searching schemes only support searching for OR of keywords or AND of two sets of keywords. Extended the types of queries to support disjunctive normal forms (DNF) of keywords. The main drawback of existing private searching schemes is that both the computation and communication costs grow linearly with the number of users executing queries. When applying these schemes to a large-scale cloud environment, querying costs will be extensive.

The previous work was the first to make private searching techniques applicable to a cloud environment. However, requires the cloud to return all of the matched files, which may cause a waste of bandwidth when only a small percentage of files are of interest. To solve the problem, introduced the concept of differential query services. The main difference between this work and that provide two extensions to address different aspects of the problem, and conduct extensive experiments on a real cloud to verify the effectiveness of the proposed schemes.

III. SYSTEM ARCHITECTURE

The aggregation and distribution layer (ADL), many users, and the cloud, as shown in Fig. 1. For ease of explanation, in this paper use only one ADL, but multiple ADLs can be deployed as necessary. An ADL is deployed in an organization that authorizes its staff to share data in the cloud. The staff members, as the authorized users, send their queries to the ADL, which will aggregate user queries and send a combined query to the cloud. Then, the cloud processes the combined query on the file collection and returns a buffer that contains all of matched files to the ADL, which will distribute the search results to each user. To aggregate sufficient queries, the organization may require the ADL to wait for a period of time before running our schemes, which may incur a certain querying delay. In the supplementary file, In the computation and communication costs as well as the querying delay incurred on the ADL. To further reduce the communication cost, a differential query service is provided by allowing each user to retrieve matched files on demand. Specifically, a user selects a particular rank for his query to determine the percentage of matched files to be returned. This feature is useful when there are a lot of files that match a user's query, but the user only needs a small subset of them.

The ADL is deployed inside the security boundary of an organization, and thus it is assumed to be trusted by all of the users. In the supplementary file the EIRQ schemes work without such an assumption.

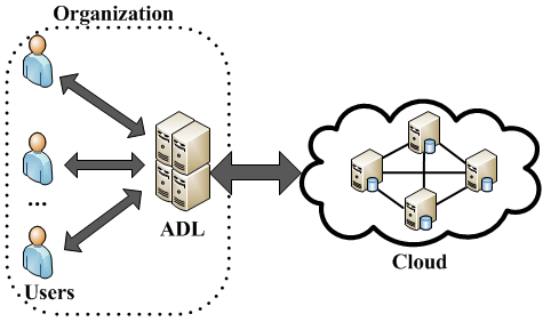


Fig. 1. System model.

The communication channels are assumed to be secured under existing security protocols, such as SSL, during information transfer. With these assumptions, as long as the ADL obeys our schemes, a user cannot know anything about other user's interests, and thus the cloud is the only attacker in our security model. As in existing work the cloud is assumed to be honest but curious. That is, it will obey our schemes, but still wants to know some additional information about user privacy. It classified user privacy into search privacy and access privacy. In our work, user queries are classified into multiple ranks, and thus a new kind of user privacy, rank privacy, also needs to be protected against the cloud. Rank privacy entails hiding the rank of each user query from the cloud, i.e., the cloud provides differential query services without knowing which level of service is chosen by the user. Rank privacy can be classified into basic level and high level, where basic level will hide the rank of each query from the cloud, and the high level will further hide the number of ranks from the cloud. Our design goal can be subdivided as follows:

• **Cost efficiency.** The users can retrieve matched files on demand to further reduce the communication costs incurred on the cloud.

• **User privacy.** The cloud cannot know anything about the user's search privacy, access privacy, and at least the basic level of rank privacy.

IV **Overview of the Ostrovsky Scheme**

Introduce the Ostrovsky scheme which relies on a public key cryptosystem, the Paillier cryptosystem. Let Epk(m) denote the encryption of plaintext m under public key pk. The Paillier cryptosystem has the following homomorphic properties:

• $Epk(a) \cdot Epk(b) = Epk(a + b)$

• $Epk(a)b = Epk(a \cdot b)$

The Paillier cryptosystem allows the performance of certain operations, such as multiplication and exponentiation, on cipher text directly. Given the resultant cipher text, the user can obtain the corresponding plaintext that processes addition and multiplication operations. The Ostrovsky scheme consists of three algorithms, the working process of which is shown in Fig. 2-(a). Two assumptions are used in their scheme: first, a dictionary that consists of the universal keywords is assumed to be publicly available; second, the users are assumed to have the ability to estimate the number of files that match their queries. To better illustrate its working process, we provide an example in the supplementary file.

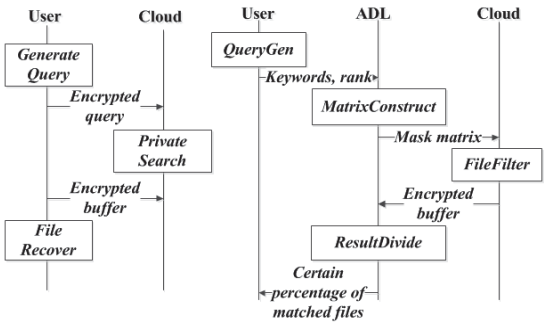
Step 1.

The user runs the Generate Query algorithm to send an encrypted query to the cloud. The query is a bit string encrypted under the user's public key, where each bit is an encryption of 1, if the keyword in the dictionary is chosen otherwise, it is an encryption of 0.

Step 2.

The cloud runs the Private Search algorithm to return an encrypted buffer to the user. The cloud processes the encrypted query on every file in the collection to generate an encrypted c-e pair, and maps it to multiple entries of an encrypted buffer. For file Fj , the corresponding c-e pair, denoted as (cj, ej), is generated as follows: the bits in query Q corresponding to keywords in Fj are multiplied together to form cj = $\pi_{Dic[i] \in Fj} Q[i]$, where Dic[i] denotes the i-th keyword in the dictionary, and file content |Fj | is powered to cj to form ej = c |Fj | j . Then, the cloud constructs a buffer of size β. Let B denote the buffer, where the i-th entry, denoted as B[i], consists of two parts, denoted as B[i, 1] and B[i, 2], both of which are initialized with an encryption of 0 under the user's public key. To map (cj, ej) to the buffer, the cloud randomly chooses an entry, say p∗, and multiplies (cj, ej) to this entry by performing B[p∗, 1] = B[p∗, 1] · cj and B[p∗, 2] = B[p∗, 2] · ej . The mapping operation will be performed γ times. After mapping all pairs to the buffer, each buffer entry has one of the three statuses: survival, collision, and mismatch. If only one matched file

is mapped, the entry state is survival; if more than one matched file is mapped, the entry state is collision; if no matched files are mapped, the entry state is mismatch.



(a) The Ostrovsky scheme (b) The EIRQ-Efficient scheme

Step 3.

The user runs the File Recover algorithm to recover files. The user decrypts the buffer, entry by entry, to obtain the plaintext c-e pairs. For the entries in the survival state, file content can be recovered by dividing the plaintext e value by the plaintext c value. The security of the Ostrovsky scheme derives from the semantic security of the Paillier cryptosystem. The key

technique of their scheme is that the files mismatching a user's query are processed to encrypted 0s, which have no impact on the matched files, even if they are mapped in the same entry. Thus, the buffer size only depends on the number of matched files, which is much smaller than the number of files stored in the cloud.

IV EFFICIENT INFORMATION RETRIEVAL FOR RANKED QUERY (EIRQ)

Two fundamental problems should be resolved: Firstly, It should determine the relationship between query rank and the percentage of matched files to be returned. Suppose that queries are classified into 0 ~ r ranks. Rank-0 queries have the highest rank and Rank-r queries have the lowest rank. In this paper, It determine this relationship by allowing Rank-i queries to retrieve (1 − i/r) percent of matched files. Therefore, Rank-0 queries can retrieve 100% of matched files, and Rank-r queries cannot retrieve any files. Secondly, it should determine which matched files will be returned and which will not. In this paper, simply determine the probability of a file being returned by the highest rank of queries matching this file. Specifically, first rank each keyword by the highest rank of queries choosing it, and then rank each file by the highest rank of its keywords. If the file rank is i, then the probability 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. It will illustrate how to adjust the buffer size and mapping times to achieve this goal. EIRQ-Efficient mainly consists of four algorithms, Since algorithms Query Gen and Result Divide are easily understood, provide the details of algorithms Matrix Construct and File Filter .

Step 1.

The user runs the Query Gen algorithm to send keywords and the rank of the query to the ADL. Since the ADL is assumed to be a trusted third party, this query will be sent without encryption.

Step 2.

After aggregating enough user queries, the ADL runs the Matrix Construct algorithm to send a mask matrix to the cloud. The mask matrix M is a d-row and r-column matrix, where d is the number of keywords in the dictionary, and r is the lowest query rank. Let M[i, j] denote the element in the i-th row and the j-th column, and let l be the highest rank of queries that choose the i-th keyword Dic[i] in the dictionary. M is constructed as follows: for the i-th row of M that corresponds to Dic[i], M[i, 1], . . . , M[i, r − l] are set to 1, and M[i, r − l + 1], . . . , M[i, r] are set to 0, then each element is encrypted under the ADL's public key pk. For the

rows that correspond to Rank-l keywords, the ADL sets the first r − l elements, rather than random r − l elements, to 1. The reason is to ensure that, given any Rank-l file Fj , when choose a random number k, the probability of all of the k-th elements of the rows that correspond Fj's keywords being 0 is l/r, which is determined by the highest rank of Fj keywords.

Step 3.

The cloud runs the File Filter algorithm to return a buffer that contains a certain percentage of matched files to the ADL. Specifically, the cloud multiplies the k-th elements of the rows that correspond to Fj keywords together to form cj , where k = j mod r. Then, it powers | Fj | to cj to obtain ej , and maps the c-e pair into multiple entries of a buffer, as in the Ostrovsky scheme. Note that, with Step 2, make sure that, for a Rank l file Fj , the probability of cj being 0 is l/r, and thus the probability of Fj being filtered out is l/r.

Step 4.

The ADL runs the Result Divide algorithm to distribute search results to each user. File contents are recovered as the File Recover algorithm in the Ostrovsky scheme. To allow the ADL to distribute files correctly, the cloud to attach keywords to the file content. Thus, the ADL can find out all of the files that match users' queries by executing keyword searches.

V. **Security Analysis**

The EIRQ schemes can provide search privacy, access privacy, and rank privacy as follows.

Search Privacy

In the three schemes, the combined query sent to the cloud is encrypted under the ADL's public key with the Paillier cryptosystem. The query is a matrix of encrypted 0s and 1s. The Paillier cryptosystem is semantically secure, and the cipher text of every 1 or 0 is different from other 1s or 0s. Therefore, the cloud cannot deduce what each user is searching for from the encrypted query.

Access Privacy

In the three schemes, the cloud processes the encrypted query on each file in a collection, and maps the processing result into a buffer, which is encrypted with the ADL's public key. The cloud conducts this process for all files in the same way. Therefore, the cloud cannot know which files are actually returned from the encrypted buffer.

Rank Privacy

In EIRQ-Simple, the messages from the ADL to the cloud are r encrypted queries, the buffer size, and the mapping times, where r is the information, Given r, the cloud only knows the number of query ranks without knowing how many users are in each rank, nor which users are in which ranks. Therefore, EIRQ-Simple can protect the basic level of rank privacy for a user. In EIRQ-Privacy, the message from the ADL to the cloud is a d-row and mcolumn mask matrix, where d is the number of keywords in the dictionary, and m = maxγi is the maximal value of mapping times. Here, no extra information is leaked more than [3]. Therefore, EIRQ-Privacy provides a high level of user rank privacy. In EIRQ-Efficient, the message from the ADL to the cloud is a d-row and r-column mask matrix, where d is the number of keywords in the dictionary, and r is the lowest rank of user queries.

VI. EIRQ MODEL PERFORMANCE ANALYSIS

Compare EIRQ schemes from the following aspects: file survival rate and computation/ communication cost incurred on the cloud.

A. **File Survival Rate**

The queries are classified into 0 ~ 4 ranks, queries in Rank-0, Rank-1, Rank-2, Rank-3, and Rank-4 should retrieve 100%, 75%, 50%, 25%, 0% of matched files, respectively. The real failure rate in EIRQ-Simple and EIRQ-Privacy under the Ostrovsky parameter setting is much lower than i/r, and thus, the real file survival rate is higher than the desired value of 1 − i/r (about 25% and 50% of files are redundantly returned to users); Only EIRQ-Efficient, which filters a certain percentage of matched files before mapping them to a buffer, provides differential query services. Under the Bloom filter parameter setting, It obtain corresponding mapping times. Specifically, for file survival rate 100%, 75%, 50%, 25%. Based on these values, the buffer size can be calculated for different schemes. In practice, γ and β must be integers. Thus, in this scheme use γ and β to replace the corresponding values. Using these parameters, the file survival rates for different where three EIRQ schemes can provide differential query services, and no bandwidth is wasted in each EIRQ scheme. Therefore, in terms of file survival rate, the Bloom filter parameter setting can achieve better performance than the Ostrovsky parameter setting.

B. Computational Cost

The computational cost is mainly determined by the number of exponentiations performed by the cloud, which is almost the same under the Bloom filter and the Ostrovsky parameter settings. In order to justify the analyses, compare the computational cost between No Rank

and three EIRQ schemes. The comparisons of computational cost on the cloud where the number of queries in each rank ranges from 1 to 25. Under the Bloom filter parameter setting, the computational cost is approximately 14.807s in No Rank, 59.274s in EIRQ Simple, 101.075s in EIRQ-Privacy, and 14.861s in EIRQ Efficient. In the Ostrovsky parameter setting, the computational cost approximately ranges from 14.8270s to 14.8788s in No Rank, from 59.1671s to 59.3838s in EIRQ-Simple, from 114.0475s to 176.5107s in EIRQ-Privacy, and from 14.8664s to 14.9269s in EIRQ Efficient. In both settings, EIRQ-Privacy consumes the most computation cost, and EIRQ-Efficient, like No Rank, consumes the least computation cost.

VII. CONCLUSION

EIRQ schemes based on an ADL to provide differential query services while protecting user privacy. By using our 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, it simply determine the rank of each file by the highest rank of queries it matches. For our future work, try to design a flexible ranking mechanism for the EIRQ schemes.

REFERENCES

1. Fang Hao, Murali Kodialam, T. V. Lakshman and Haoyu Song(2012) “Fast Dynamic Multiple-Set Membership Testing Using Combinatorial Bloom Filters” IEEE Transactions on networking, vol 20, no.1, pp-295-304.
2. Junbeom Hur(2013) “Improving Security and Efficiency in Attribute-Based Data Sharing” IEEE Transactions On Knowledge And Data Engineering, vol. 25, no.10, pp-2271-2282.
3. Michael Mitzenmacher(2012) “Compressed Bloom Filters” IEEE Transactions on networking, vol 10, no.5, pp-604-612.
4. Ning Cao, Cong Wang, Ming Liy, Kui Ren and Wenjing Lou “Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data” IEEE Transactions On Parallel and Distributed Systems, vol 20, no.10, pp-1-11.
5. Qin Liu, Chiu C. Tan, Jie Wu and Fellow (2013) “Towards Differential Query Services in Cost-Efficient Clouds” IEEE Transactions On Parallel and Distributed Systems, vol. 20, no.10, pp-1-11.
6. Shucheng Yu, Cong Wang, Kui Ren† and Wenjing Lou “Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing” IEEE Transactions On Parallel and Distributed Systems, vol. 20, no.8, pp-1-9.
7. Xun Yi, Elisa Bertino, Jaideep Vaidya, and Chaoping Xing “Private Searching on Streaming Data Based on Keyword Frequency” IEEE Transactions On Dependable And Secure Computing” vol 20, no.5, pp-1-14.