IJDCST @June-July-2015, Issue- V-3, I-5, SW-19
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

A Study on Differential Query Services in Cost–Efficient Clouds

L.Jyothsna[1], M. Samba Siva Rao[2]
[1]M.Tech (CSE), Usharama College of Engineering and Technology, A.P., India.
[2]Assistant Professor, Dept. of Computer Science & Engineering, Usharama College of Engineering and Technology, A.P., India.

Abstract — In the simplest terms, cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. The cloud is just a metaphor for the Internet. Now a days 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 paper, i am address two fundamental issues in such an environment: privacy and efficiency. My first review a private keyword-based file retrieval scheme that was originally proposed by Ostrovsky. Their 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. In this paper, present three efficient information retrieval for ranked query (EIRQ) schemes 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.

Keywords — Cloud computing, cost efficiency, differential query services, privacy

I Introduction

The goal of cloud computing is to apply traditional super-computing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.Cloud computing as an emerging technology is expected to reshape information technology processes in the near future [1]. 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 [2]. 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. While this does provide the necessary privacy, the communication cost is high.

Private searching was proposed by Ostrovsky et al. [3], [4] (referred to as the Ostrovsky scheme in this paper), 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 homomorphic encryption) on every file in a collection. Otherwise, the cloud will learn that certain files, without processing, are of no interest to the user. 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 argue that subsequently proposed improvements, like [5], [6], also have the same drawback. 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. Solutions that incur excessive computation and communication costs are unacceptable to customers.

To make private searching applicable in a cloud environment, our previous work [7] 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.

In this paper, I am introducing 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. To illustrate, let us assume that Alice wants to retrieve 2 percent of the files that contain keywords ‘‘A, B’’, and Bob wants to retrieve 20 percent 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 Ostrovsky scheme, the cloud will have to return 2,000 files. In the COPS scheme, the cloud will have to return 1,000 files. In our scheme, the cloud only needs to return 200 files. Therefore, by allowing the users to retrieve matched files on demand, the bandwidth consumed in the cloud can be largely reduced.

Motivated by this goal, propose a 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. Focusing on different design goals, provide 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. Propose three 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. 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

Our work aims 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 [8], [9], [10], [11]. Unlike searchable encryption [12], where the user conducts searches on encrypted data, private searching performs keyword-based searches on unencrypted data. Private searching was first proposed in [3], [4], which allow a server to filter streaming data without compromising user privacy. Their solution requires the server to return a buffer of size O(flog(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 [13], the files that mismatch a query will not survive in the buffer, but the matched files enjoy a high survival rate.

Among various extensions, [5], [6] 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. Reference [8] presented an efficient decoding mechanism which allows the recovery of files that collide in a buffer position. Reference [9] proposed a recursive extraction mechanism, which requires a buffer of size O(f) when f files match a user's query. Reference [10] 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. Reference [11] 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. Thus, when applying these schemes to a large-scale cloud environment, querying costs will be extensive.

III System Model

The system mainly consists of three entities:1 the aggregation and distribution layer (ADL), many users, and the cloud, as shown in Fig. 1. For ease of explanation, only use a single ADL in this paper, 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. 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.

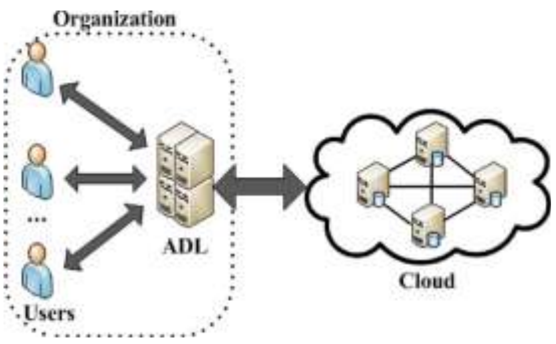


Fig. 1: System model.

Overview of the Ostrovsky Scheme

The Paillier cryptosystem allows the performance of certain operations, such as multiplication and exponentiation, on ciphertext directly. Given the resultant ciphertext, 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, provide an example in the supplementary file available online

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. Generally speaking, 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. The mapping operation will be performed gamma 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 aremapped, the entry state is mismatch.

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.

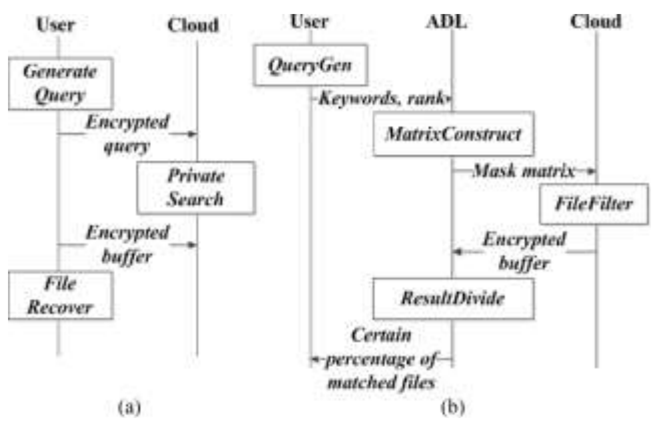


Fig. 2: Working process. (a) Ostrovsky scheme. (b) EIRQ- Efficient scheme.

The basic idea of EIQR-Efficient is to construct a privacy-preserving mask matrix with which the cloud can filter out a certain percentage of matched files before mapping them to a buffer. As proven in the Ostrovsky scheme, the file survival rate is determined by the buffer size β and mapping times γ. therefore, the basic idea of two extensions is that, for each rank i € { 0,. . . , r}, the ADL adjusts the buffer size _i and the mapping times _i to make the file survival rate q i approach 1 – i/r.

EIRQ-Efficient Scheme

Firstly, 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, simply determine this relationship by allowing Rank-i queries to retrieve 1 – i/r percent of matched files. Therefore, Rank-0 queries can retrieve 100 percent of matched files, and Rank-r queries cannot retrieve any files.

Secondly, should determine which matched files will be returned and which will not. In this paper, I am 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.

Since algorithms QueryGen and ResultDivide are easily understood, I am only provided the details of algorithms Matrix- Construct and File Filter in Alg. 1.

Algorithm 1 The EIRQ-Efficient scheme.

```
MatrixConstruct (run by the ADL with public key pk)
for $i = 1$ to $d$ do
    set $l$ to be the highest rank of queries choosing $Dic[i]$
    for $j = 1$ to $r$ do
        if $j \leq r - l$ then
            $M[i, j] = E_{pk}(1)$
        else
            $M[i, j] = E_{pk}(0)$
adjust $\gamma$ and $\beta$ so that file survival rate is 1
FileFilter (run by the cloud)
for each file $F_j$ stored in the cloud do
    for $i = 1$ to $d$ do
        $k = j \bmod r$; $c_j = \prod_{Dic[i] \in F_j} M[i, k]$; $e_j = c_j^{|F_j|}$
        map $(c_j, e_j)$ $\gamma$ times to a buffer of size $\beta$
```

Step 1. The user runs the QueryGen 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 my 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's 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's keywords together to form cj, where k ¼ j mod r. Then, it powers jFjj 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, i can 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.

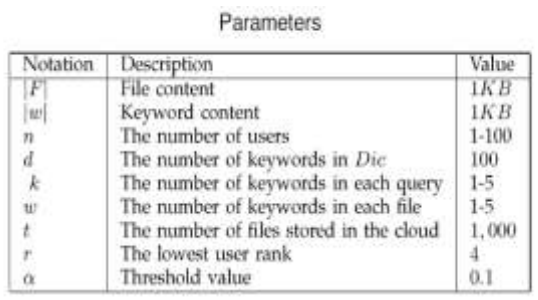
Step 4. The ADL runs the ResultDivide algorithm to distribute search results to each user. File contents are recovered as the File Recover algorithm in the Ostrovsky scheme. To allow the ADL to distribute files correctly, i require the cloud to attach keywords to the file content. Thus, the ADL can find out all of the files that match users' queries by executing keyword searches.

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, which leak more than [3]. Given r, the cloud only knows the number of queryranks without knowing how many users is in each rank, nor which users are in which ranks. Therefore, EIRQSimple 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 m-column mask matrix, where d is the number of keywords in the dictionary, and m =max γi is the maximal value of mapping times. Therefore, EIRQ-Efficient can protect the basic level of rank privacy for a user. I will evaluate the consumed energy overhead in the cloud to verify the effectiveness of our schemes. I use No Rank to denote unranked queries under the ADL. The summary of the experiment parameters are shown in below Table.


Parameters

Notation	Description	Value
$\|F\|$	File content	$1KB$
$\|w\|$	Keyword content	$1KB$
n	The number of users	1-100
d	The number of keywords in Dic	100
k	The number of keywords in each query	1-5
w	The number of keywords in each file	1-5
t	The number of files stored in the cloud	$1,000$
r	The lowest user rank	4
α	Threshold value	0.1

.

Table 1: **TOWARDS DIFFERENTIAL QUERY SERVICES IN COST-EFFICIENT CLOUDS**

IV Conclusion

In this paper, proposed three 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, simply determine the rank of each file by the highest rank of queries it matches. For our future work, will try to design a flexible ranking mechanism for the EIRQ schemes.

References

[1] P. Mell and T. Grance, ‘‘The NIST Definition of Cloud Computing (Draft),’’ in NIST Special Publication. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2011.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, ‘‘Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions,’’ in Proc. ACM CCS, 2006, pp. 79-88.

[3] R. Ostrovsky and W. Skeith, ‘‘Private Searching on Streaming Data,’’ in Proc. CRYPTO, 2005, pp. 233-240.

[4] R. Ostrovsky and W. Skeith, ‘‘Private Searching on Streaming Data,’’ J. Cryptol., vol. 20, no. 4, pp. 397-430, Oct. 2007.

[5] J. Bethencourt, D. Song, and B. Waters, ‘‘New Constructions and Practical Applications for Private Stream Searching,’’ in Proc. IEEE SP, 2006, pp. 1-6.

[6] J. Bethencourt, D. Song, and B. Waters, ‘‘New Techniques for Private Stream Searching,’’ ACM Trans. Inf. Syst. Security, vol. 12, no. 3, p. 16, Jan. 2009.

[7] Q. Liu, C. Tan, J. Wu, and G. Wang, ‘‘Cooperative Private Searching in Clouds,’’ J. Parallel Distrib. Comput., vol. 72, no. 8, pp. 1019-1031, Aug. 2012.

[8] G. Danezis and C. Diaz, ‘‘Improving the Decoding Efficiency of Private Search,’’ Int’l Assoc. Cryptol. Res., IACR Eprint Archive No. 024, Schloss Dagstuhl, Germany, 2006.

[9] G. Danezis and C. Diaz, ‘‘Space-Efficient Private Search with Applications to Rateless Codes,’’ in Proc. Financial Cryptogr. Data Security, 2007, pp. 148-162.

[10] M. Finiasz and K. Ramchandran, ‘‘Private Stream Search at the Same Communication Cost as a Regular Search: Role of LDPC Codes,’’ in Proc. IEEE ISIT, 2012, pp. 2556-2560.

[11] X. Yi and E. Bertino, ‘‘Private Searching for Single and Conjunctive Keywords on Streaming

Data,’’ in Proc. ACM Workshop Privacy Electron. Soc., 2011, pp. 153-158.

[12] B. Hore, E.-C. Chang, M.H. Diallo, and S. Mehrotra, ‘Indexing Encrypted Documents for Supporting Efficient Keyword Search,’’ in Proc. Secure Data Manage., 2012, pp. 93-110.

[13] P. Paillier, ‘‘Public-Key Cryptosystems Based on Composite Degree Residuosity Classes,’’ in Proc. EUROCRYPT, 1999, pp. 223-238.