A comparative study on the algorithms for data privacy and security in cloud computing

Syed Shafaque Fatma, Meera Narvekar

Abstract— Cloud computing is a technology expected to redefine the advances in information technology. In data privacy protection and retrieval- control is one of the most challenging research works in cloud computing. Cloud computing offers an innovative business model for organizations with minimal investment. Security is one of the major issues which affect the growth of cloud. An important problem in this environment is to protect the user privacy while querying the data from the cloud; to address this researcher have developed many techniques, but the problem with those techniques is the heavy computational and bandwidth costs which are unacceptable to the users. This paper gives an overview of some schemes which address the problem of bandwidth and computational costs.

Index Terms— cloud computing, user privacy, encryption, ADL, mask matrix.

1 INTRODUCTION

Cloud computing is an emerging technology which is being used widely these days. Due to the advantages of cloud computing, e.g., cost-effectiveness, flexibility and scalability, more and more organizations are now opting to outsource their data for sharing in the cloud. In a cloud application, an organization subscribes the cloud services and gives access to its staff to share files in the cloud. Each file is described by some keywords, and the staff, as authorized users, can retrieve files which they are interested in by querying the cloud with those keywords. In such a scenario, protection of user privacy from the cloud, which is outside the security boundary of the organization, becomes a key problem. User privacy can be classified into search privacy and access privacy [6]. 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.

2 RELATED WORK

For the private searching in the cloud many algorithms were proposed. Private searching is proposed by [1], where the data is stored in the clear form, and the query is encrypted with the Paillier cryptosystem. The cloud stores all files into a compact buffer, with which the user can successfully recover all wanted files with high probability. In the following work, [2] reduced the communication cost in [1] by solving a set of linear programs; [7] presented an efficient decoding mechanism for private searching. The main drawback of the current private searching techniques is that both the computation and communication costs grow linearly with the number of users that are executing searches. Thus, when applying these schemes to a large-scale cloud environment, querying costs will be extensive. Ranked searchable encryption enables users to retrieve the most matched files from the cloud in the case that both the query and data are in the encrypted form. The work by [8], which only supports single-keyword searches, encrypts files and queries with Order Preserving Symmetric Encryption (OPSE) [9] and utilizes keyword frequency to rank results. Their following work [10], which supports multiple-keyword searches, uses the secure KNN technique [11] to rank results based on inner products. The main limitation of these approaches is that user access privacy [6] will not be preserved. Let us now see the three algorithms in detail i.e. Ostrovsky scheme, COPS protocol, EIRQ scheme. All these algorithms address to the private searching in the cloud environment.

2.1 Ostrovsky scheme

A key privacy search solution was proposed by Ostrovsky et al. [1], which can provide the same privacy level as downloading the entire database from the cloud with significantly less communication costs. By asking the cloud to return the entire database, the cloud cannot know which files are really interested by a user. However, the Ostrovsky scheme has a high computation cost, since it must require the cloud to process the encrypted query on every file present in the database; Otherwise, the cloud will come to know that certain files are not related to that user's query. Therefore, it will become a performance bottleneck when the cloud needs to process thousands of files. Ostrovsky gave a scheme based on the homomorphism of the Paillier cryptosystem providing this capability. First, a public dictionary of keywords D is fixed. To construct a query for the disjunction1 of some keywords K is a subset of D, the user produces an array of cipher texts, one for each $w \in D$. If $w \in K$, a one is encrypted; otherwise a zero is encrypted. A server processing a document in its stream may then compute the product of the query array entries corresponding to the keywords found in the document. This will result in the encryption of some value c, which, by the homomorphism, is nonzero if and only if the document matches the query. The server may then in turn compute E(c) f = E(cf), where f is the content of the document, obtaining either an encryption of (a multiple of) the document or an encryption of zero. Ostrovsky and Skeith propose the server keep a large array of cipher texts as a buffer to accumulate matching documents; each E(cf) value is multiplied into a number of random locations in the buffer. If the document matches the query then c is nonzero and copies of that document will be placed into these random locations; otherwise, c = 0 and this step will add an encryption of 0 to each location, having no effect on the corresponding plaintexts. A fundamental property of their solution is that if two different matching docu-

ments are ever added to the same buffer location, a collision will result and both copies will be lost. If all copies of a particular matching document are lost due to collisions then that document is lost, and when the buffer is returned to the client, they will not be able to recover it. To avoid the loss of data in this approach one must make the buffer sufficiently large so that this event does not happen. This requires that the buffer be much larger than the expected number of required documents. In particular, Ostrovsky and Skeith show that a given probability of successfully obtaining all matching documents may be obtained with a buffer of size O(m log m)^2, where m is an upper bound on the number of matching documents. While effective, this scheme results in inefficiency due to the fact that a significant portion of the buffer returned to the user consists of empty locations and document collisions [2]. Let E(m) denote the encryption of plaintext m. The Paillier cryptosystem has the following homomorphic properties: E(a) • E(b) = E(a + b) and E(a)b = E(a • b) The Ostrovsky scheme consists of three algorithms, as shown in Algorithm. We use a simple example to illustrate its working process as follows: public dictionary Dic = (A,B,C,D) and files stored in the cloud are as in Table I; A user, Alice, wishes to retrieve files with keywords "A, B". In the first step, Alice runs the Generate Query algorithm to generate a query Q = (E(1),E(1),E(0),E(0)), where each entry is an encryption of 1 if the corresponding keyword is chosen; otherwise it is 0. In the second step, the cloud runs the Private Search algorithm to generate occurrence-content pairs. For example, user keywords "A, B" appear in F1, both of which correspond to E(1) in user query Q. Thus, the occurrence of the user keywords is the product of corresponding entries in Q, i.e., c1 = E(1) • E(1) = E(1 + 1) = E(2). File content is then powered by the occurrence, i.e., e1 = c|F1| 1 = E(2 • |F1|). Then, the cloud maps each pair many times to a compact buffer format". For each buffer entry, there are three statuses: survival, collision, and mismatch, where a collision will appear only when more than one matched file is mapped, a survival will appear when only one matched file is mapped, and a mismatch will appear when unmatched files are mapped. For example, in Fig. 3, the second entry is a collision. When a collision happens, no files in the entry can be recovered. In the third step, Alice runs the File Recover algorithm to recover files. Note that if a file is mismatching Q, then the occurrence is an encryption of 0, and the file content is processed to be an encryption of 0. Otherwise, the occurrence is an encryption of some value v larger than 0, and the file content is processed to be an encryption of v • |Fj |. Therefore, the user can obtain file content by dividing the content by the occurrence. This scheme also provides a collision-detection mechanism to let the user get rid of the conflicting copies [2].

2.2 COPS protocol

The COPS (Cooperate private searching) protocol reduces the communication and computational cost by introducing the concept of aggregation and distribution layer (ADL). It is deployed in the organization and it works between the users and cloud. It combines the multiple user queries before sending it to the cloud. This reduces the communication cost as the different users can ask for the same file and if that query is sent twice then the cost incurred will be more. Hence, if the query is combined then in one request the file can be retrieved. To illustrate, let us assume that files F1, F2, and F3, which are stored in the cloud, are described with keywords "A, B", "B", and "C", and Alice and Bob query data with "A, B" and "A, C", respectively. Under the ADL, the cloud needs to execute the query only once to return F1, F2, and F3 to the ADL. Compared to the Ostrovsky scheme, the computation and communication costs are saved by 50% and 25%, respectively. Note that introducing the ADL will incur some processing delay for aggregating queries. However, the degree of aggregation can be controlled through a time-out mechanism to meet a given processing delay requirement. When the time-out is set to zero, this is degraded to the normal sequential search [4].

2.3 EIRQ scheme

The basic idea of EIRQ (Efficient information retrieval using ranked queries) is that a privacy-preserving mask matrix is used to filter out a certain percentage of files before mapping them to a buffer. Before illustrating EIQR, two fundamental problems should be resolved: First, we should determine the relationship between query rank and the percentage of returned matched files. Suppose that queries are classified into r ranks, where Rank-0 queries have the highest rank and Rank-r queries have the lowest rank. In this paper, we simply determine this relationship by allowing Rank-i queries to retrieve (1−i/r) percent of statuses matched files. Therefore, Rank-0 queries can retrieve 100% of the matched files, and Rank-r queries cannot retrieve any files. Second, we should determine which matched files will be returned and which will not. In this paper, we simply determine the probability of a file being returned by the highest rank of queries matching this file. Specifically, we 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 [4].EIRQ consists of four algorithms; we will use the following example to describe its working process. The dictionary and files are the same as in table II; users are classified into four ranks, where Alice, a Rank-0 user, queries with keywords "A, B", and Bob, a Rank-1 user, uses keywords "A, C". According to our rules, "A, B" are Rank- 0 keywords, "C" is a Rank-1 keyword, and "D" is a Rank-4 keyword. Correspondingly, F1 and F2 are Rank-0 files which will be returned with a probability of 1, F3 and F4 are Rank-1 files which will be returned with a probability of 75%, and F5 is a Rank-4 file which will not be returned. First each user runs the Query Gen algorithm to send a query to the ADL, where the user query consists of the chosen keywords and the query rank. Given users' 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 highest rank of queries. The mask matrix M can be constructed as follows: For each keyword w, the ADL first sets w's rank with l, the highest query rank choosing this keyword. Then, for the row corresponding to keyword w, the ADL sets the first r − l columns to 1 and the last l columns to 0. Note that, the reason for setting the first r − l columns, rather than random r − l columns, to 1 is to ensure that, given any two files with rank l, the probability of the product of the columns corresponding to file keywords being 0 is l/r. Based on the mask matrix, the cloud runs the File Filter algorithm to filter out the percentage of matching files based on the user query rank and returns a union buffer to the ADL. The process is as follows: For each file

Fj , the cloud first multiplies the k-th columns that correspond to Fj 's keywords in the mask matrix to obtain cj , where k = j mod r. Then, the cloud powers the file content to cj to obtain ej and maps (ci, ei) to many entries of a union buffer as the Ostrovsky scheme. Here, cj denotes the occurrence of ranked keywords in file Fj. Thus, cj will be larger than 0, and file Fj will be returned only when l + k ≤ r, where k = j mod r. The ADL then runs the Result Divide algorithm to distribute files to each user. The ADL first recovers all files that match user queries as the File Recover algorithm in the Ostrovsky scheme. Then, the ADL distributes appropriate files to each user based on the user queries. To make sure that the ADL distributes files correctly, we can require the cloud to attach file keywords with the file content. Thus, the ADL can find out all of the files that match each user's query by executing keyword searches [4].

3 COMPARISON

The following tables shows the comparison of the Ostrovsky scheme, COPS protocol and EIRQ scheme with respect to various parameters such as security, computational and communicational cost.

4 CONCLUSION

TABLE 1
COMPARISON

Schemes	Parameter		
	Security	***Computational cost***	***Bandwidth cost***
Otrovsky	Yes	No	No
COPS protocol	Yes	Yes, to some extent	Yes, to some extent
EIRQ	Yes	Yes	Yes

Cloud computing is used for sharing and retrieving information. However, while retrieving information from cloud environment it is necessary to get desired information with optimal communication and computation cost. In this paper, we have analyzed various algorithms is which is used for efficient information retrieval in cloud environment. We have also shown the comparison of these algorithms which is useful for better understanding of these algorithms in terms of different parameters.

REFERENCES

[1] R. Ostrovsky and W. Skeith III, "Private searching on streaming data," in Proc. of ACM CRYPTO, 2005.

[2] J. Bethencourt, D. Song, and B. Waters, "New techniques for private stream searching," ACM Transactions on Information and System Security, 2009.

[3] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Cooperative private searching in clouds" http://www.cis.temple.edu/ cctan/TR1.pdf, Tech. Rep., 2011.9.

[4] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in Proc. of IEEE INFOCOM, 2012.

[5] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," NIST Special Publication, 2011.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in Proc. of ACM CCS, 2006.

[7] G. Danezis and C. Diaz, "Improving the decoding efficiency of private search," in IACR Eprint archive number 024, 2006.

[8] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in Proc. of IEEE ICDCS, 2010

[9] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," Advances in Cryptology-EUROCRYPT, 2009.

[10] Lou, "Privacy-preserving multikeyword ranked search over encrypted cloud data," in Proc. Of IEEE INFOCOM, 2011.

[11] W. Wong, D. Cheung, B. Kao, and N.Mamoulis, "Secure knn computation on encrypted databases," in Proc. of ACM SIGMOD, 2009.