Private Range Queries on Outsourced Databases

Lu Li[1](✉), Liusheng Huang[1], An Liu[2], Yao Shen[1], Wei Yang[1], and Shengnan Shao[3]

[1] School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
liluzq@mail.ustc.edu.cn

[2] School of Computer Science, Soochow University, Suzhou, China

[3] School of Software Engineering of USTC, Suzhou, China

Abstract. With the advent of cloud computing, data owners could upload their databases to the cloud service provider to relief the burden of data storage and management. To protect sensitive data from the cloud, the data owner could publish an encrypted version of the original data. However, this will make data utilization much harder. In this paper, we consider the problem of private range query. Specifically, the data owner wants to obtain all the data within a query region, while keeping the query private to the service provider. Previous works only provide partial security guarantee, and are inefficient to deal with large scale datasets. To solve this problem, we present a fully secure scheme based on private information retrieval (PIR) and batch codes (BC).

Keywords: Secure range query · Cloud computing · Private information retrieval

1 Introduction

In cloud computing paradigm, data owners could outsource their databases to the service provider, and thus reap huge benefits from releasing the heavy storage and management tasks to the cloud server. However, sensitive data, such as medical or financial records, should be encrypted before being uploaded to the cloud [7]. To preserve the privacy of sensitive data, decryption keys should never be revealed to the cloud server. Unfortunately, this will introduce new challenges to data utilization. Considering the multi-dimensional range queries, which are typical database query operations in real life, lightweight encryption scheme, e.g. block cipher [3], cannot be directly applied for the server to conduct the queries. To enable private queries over encrypted data, Lu [1] proposed **LSE**, a symmetric encryption scheme, which can deal with the private single-dimensional range queries in *logarithmic* time. As a multi-dimensional range query can be decomposed to several single-dimensional queries, **LSE** theoretically can be extended to support private multi-dimensional range queries. However, the extended scheme will cause significant information leakage. Specifically, the cloud server will learn the exact relationship between every single-dimension of the query data and the outsourced data. Other recent works, e.g. [2], are also suffered from leaking this kind of information to the cloud server.

To prevent the single-dimensional information leakage, Wang *et al.* [4] proposed **Maple** by leveraging Hidden Vector Encryption [5] in a novel way. Although **Maple** can provide stronger privacy guarantee than previous works, it is still not fully secure in the sense of cryptography, e.g., the cloud server can learn the query path as well as which data records are the query results during each query. The cloud server can easily determine the data distribution based on this knowledge through statistical analysis. Besides, this work is inefficient due to the heavy computational operations of cryptography.

To deal with this problem, we present in this paper a fully secure scheme based on private information retrieval (PIR) [8] and batch codes (BC) [6].

2 Approach

To improve computational efficiency and to reduce storage cost, we adopt the block cipher as the underlying encryption scheme. Specifically, we use the AES in counter mode to encrypt the data records before uploading them to the cloud server.

2.1 Basic Scheme

We first give a basic private range query protocol. When the data owner wants to issue a range query, she does not send the server any information about the query region. Instead, she just submits a query requirement. Once receiving a query requirement from the data owner, the cloud server forwards all the encrypted nodes and the topology structure in high levels of the encrypted tree to the data owner. Each node in the bottom level is required to be associated with an identity that contains the indexes of its children nodes in the next level. The data owner first decrypts the root node, and then decides which nodes in the next level should be selected based on the intersection judgement. This process will continue until the data owner obtains the desired indexes of the children nodes of the bottom level. The above process is secure, because the data owner does not send any confidential information to the cloud server. Also, storage and computational overhead can be reduced by restricting the number of levels. Obviously, this process only enables the data owner to perform traversal over limited levels, as transferring the nodes in the next level will incur large communication cost and storage overhead on the data owner. Fortunately, this problem can be resolved by adopting PIR protocol. Recalling that during the above process, the data owner has already known which nodes in the next level should be transferred. In the next step, the data owner and the cloud server could engage in PIR protocol several times to let the data owner extract the information of these nodes, without revealing the required nodes to the cloud server. The data owner then decrypts these nodes and judges which children nodes should be transferred in the next level. This process could continue to the end, and the data owner will obtain all the data records within the query region.

Although the above solution ensures the data owner obtains the correct query result and does not leak much information, it may cause large volume of computation on the server side, because the computational time on the cloud server will increase linearly with respect to the amount of query result. The protocol will suffer from inefficiency in such a case. In the next subsection, we will present an improvement to greatly reduce the cost. Integrating the improvement into the above query protocol, we can easily eliminate the potential security risk by masking the amount of elements issued in each level, and it will not cause significant increase of the overhead.

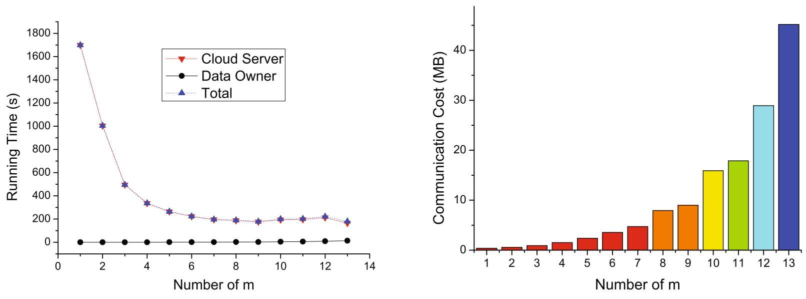
2.2 Efficient Private Range Query Protocol

As shown in [6], batch code constructions can be used to support multi-round PIR. We now give an example to show how it works. Considering a database with n bits, one can straightforwardly use the above PIR scheme twice to obtain two elements of the database. However, the corresponding computational overhead is $2n$ multiplications at the server. If we partition the database into two parts: L and R containing $n/2$ bits each, and store L on the first bucket, R on the second and $L \oplus R$ on the third. Now one can extract two elements by calling single-bit PIR in each bucket, and the computational overhead at the server is reduced to $3n/2$ multiplication. Also, the database can be partitioned into m parts, i.e. $n = n_1||...||n_m$, where $|n_i| = \lceil \frac{n}{m} \rceil, i = 1, ..., m$. We also append a part $n_{m+1} = n_1 \oplus ... \oplus n_m$. Now it is easy to know that we can retrieve any two bits in n by extracting single bit from each of the $m + 1$ parts, and the computational overhead at the server will be reduced to $(m + 1)n/m$ multiplication.

This process can be recursively applied to support 2^d-bit information retrieval, and the computational overhead at server side is monotonic with respect to m. However, we cannot simply set m the maximum value to reduce the total cost, as this will also increase the corresponding computational overhead at the data owner side and the communication overhead. Fortunately, the optimization coding scheme can be selected easily by considering the overall overhead, and using such a coding scheme will greatly reduce the total cost.

3 Experiments

Our protocols are implemented in C++ using GMP library and tested on two MacBook Pro laptops (2.2GHz CPU and 16GB RAM) connected by a 100 Mbps LAN. The default experimental setting is as follows: the encryption key length is 128, the bit length of N is 1024, the bit length l of each data record is 24, the maximum number of result points k is 64, the number of data records n is 4,000,000. We show in Fig. 1 the computation time and message volume by executing different partition scheme (where m varies from 1 to 13). Clearly, the computation overhead will decrease rapidly, and the transmitted message volume will increase slowly when m varies from 1 to 7. So, we could easily select the best encoding scheme to greatly reduce the total overhead. Based on the optimization partition scheme, even to deal with 10,000,000 data records, our protocol can be finished within 10 minutes, which is much faster than the most secure previous work (Maple requires 928 seconds to deal with 100,000 data records).



(a) Computation Complexity (b) Communication Complexity

Fig. 1. Performance evaluation v.s. number of m

4 Conclusions

In this paper, we have studied the problem of private range queries in outsourced database. We have presented a fully secure scheme based on private information retrieval. By adopting batch code, our scheme is much more efficient than state-of-the-art approaches.

References

1. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: NDSS (2012)
2. Wang, P., Ravishankar, C.V.: Secure and efficient range queries on outsourced databases using $\hat{R}$ tree. In: ICDE, pp. 314–325 (2013)
3. Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Chapman Hall/Crc Cryptography and Network Security Series). Chapman Hall/CRC (2007)
4. Wang, B., Hou, Y., Li, M., et al.: Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In: ASIACCS, pp. 111–122 (2014)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
6. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: STOC, pp. 262–271 (2004)
7. Armbrust, M., Fox, A., Griffith, R., Joseph, A., et al.: A View of Cloud Computing. Communications of the ACM **53**(4), 50–58 (2010)
8. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally private information retrieval. In: FOCS, pp. 364–373 (1997)