

A Comparison of the Query Execution Algorithms in Secure Database System

Young-Dal Jang, Ji-Hong Kim

Department of Information and Telecommunication, Semyung University, Korea

Article Info

Article history:

Received Jul 28, 2015

Revised Nov 21, 2015

Accepted Dec 2, 2015

Keyword:

Bloom Filter

Bucket Index

DAS (database as a service)

Database encryption

ABSTRACT

In accordance with the database management, DAS (database as service) model is one solution for outsourcing. However, we need some data protection mechanisms in order to maintain the database security. The most effective algorithm to secure databases from the security threat of third party attackers is to encrypt the sensitive data within the database. However, once we encrypt the sensitive data, we have difficulties in queries execution on the encrypted database. In this paper, we focus on the search process on the encrypted database. We proposed the selective tuple encryption method using Bloom Filter which could tell us the existence of the data. Finally, we compare the search performance between the proposed method and the other encryption methods we know.

Copyright © 2016 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Ji-Hong Kim,

Department of Information and Telecommunication, Semyung University,
65, Semyeong-ro, Jecheon-si, Chungcheongbuk-do, Korea.

E-mail: jhkim@semyung.ac.kr

1. INTRODUCTION

The new trend of DAS system [1] leads to a new concern, security. Especially, when the sensitive information is contained in the database, it needs its confidentiality in such a model. In a typical setting of the problem, the confidential portions of the data should be stored at a remote location in an encrypted form at all times. For example, data encryption becomes important when the client chooses to hide away certain contents from server-side entities. Two new challenges emerge: (1) How to encrypt the sensitive data. (2) How to support queries on the encrypted relational data.

In order to query effectively on the encrypted database, user query should be translated to the modified form. The modified query should be suitable to search the data in the encrypted database stored in a DB server. Therefore we use the Bloom Filter and Bucket index for the index on the encrypted database search. The metadata in client module has hash functions for generating Bloom Filter and the bucket index for the numerical attributes. Therefore the original query is converted to the modified query with the help of the metadata. The index in server part is used to search the exact data from the encrypted database.

The remainder of this paper is organized as follows. Section 2 describes the general database encryption methods. Section 3 shows the details of the proposed method and Section 4 shows the comparison between the proposed method and the other methods. Finally, we conclude with a summary and directions for future work.

2. RELATED WORK

In general, we can classify the database encryption methods according to the encryption unit. There are the element based encryption methods, the column/row based encryption methods, and the table based encryption methods.

Table 1. The comparison between various Encryption Methods

Encryption Unit	Encryption Speed	Memory Size	Search Speed	
			Keyword Search	Numeric Operation
Element	Slow	Large	High	Low
Column/Row	Medium	Medium	Medium	Medium
Table	Fast	Small	Low	Low

The element based encryption method is useful only for the search for the same valued data, but it is not good for the numeric operations. It always needs to do the element level decryption before doing numeric operation. For example, there is one plain text table such as $R(A_1, A_2, A_3, \dots, A_n)$. In this case, we store an encrypted relation $R^e(A_1^e, A_2^e, A_3^e, \dots, A_n^e)$ where each attribute A_i^e corresponds to the encrypted data for each attribute A_i .

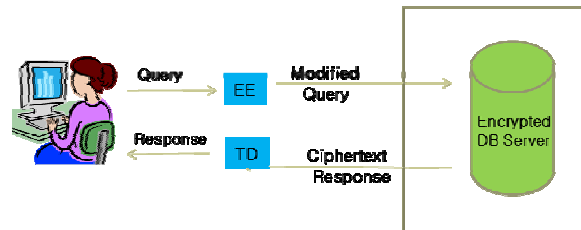


Figure 1. The element based encryption method

Table 2. The original query and the modified query

The original plain-text query	The transformed query
SELECT name	SELECT name
FROM std_info	FROM std_info(E)
WHERE city = seoul	WHERE city = seoul(E)

In Figure 1, a query generated by a database user is transformed to the modified query which could search the exact data from the encrypted database. In Table 2, if the `std_info` table is the student personal information, `std_info(E)` is the encrypted database which contains the encrypted elements in the `std_info` table. The “seoul(E)” is the encrypted form of the seoul

The column/row based encryption method is the method which encrypts the data by the unit of the column or row. Typically, the row based encryption method which is also called by tuple based encryption method, is generally used.

For each relation $R(A_1, A_2, A_3, \dots, A_n)$, we store on the server an encrypted relation $R^e(etuple, A_1^{bi}, A_2^{bi}, A_3^{bi}, \dots, A_n^{bi})$ where the *etuple* stores an encrypted string that corresponds to a tuple in relation R . Each attribute A_i^{bi} corresponds to the bucket index for the attribute A_i that will be used for query processing at the server. In this case we have to map the domain of values of attribute $R.A_i$ into partition $\{p_1, p_2, p_3, \dots, p_k\}$ such that these partitions taken together cover the whole domain; and any two partitions do not overlap. Formally, we define a function *partition* as follows; $partition(R.A_i) = \{p_1, p_2, p_3, \dots, p_k\}$. We will call k the number of the bucket. This method is proposed by Hore [2] and known to be a generally good method to protect the database these days.

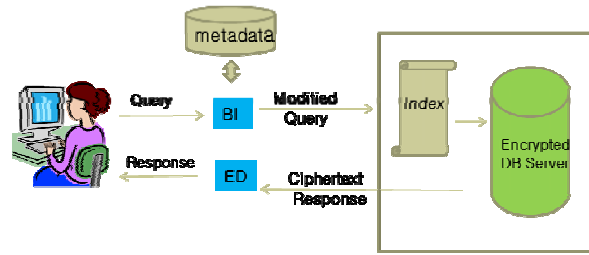


Figure 2. The typical tuple encryption method

Table 3. The original query and the modified query

The original plain-text query	The transformed query
SELECT name	SELECT etuple
FROM std_info	FROM std_info(E)
WHERE age = 30	WHERE (age) ^{bi} = age (30) ^{bi}

In Figure 2, BI means the Bucket Index used for query conversion process and ED means etuple decryption process. A query originated from a user is transformed to the modified query in order to search the exact data from the encrypted database. The metadata store the bucket index of the each attribute. So, the value of the attribute involved in the user query is converted to the bucket index. Then the bucket index value is inserted into the modified query which could search the exact data from the tuple encrypted database. The query result returned from the server is the encrypted string *etuple*. The client module should decrypt *etuple* and extract the right result from the decrypted data.

In Table 3, the *std_info* table contains the plaintext form of the student personal information and *std_info(E)* is the encrypted database which contains the encrypted string of each tuple. The age^{bi} is the bucket index value of the age element.

Finally, table based encryption method, is used to make the backup data in order to store data on the original database periodically. Therefore, this method is not good for query processing on the encrypted database.

3. THE PROPOSED DATABASE ENCRYPTION MODEL

Even if some sensitive data exists in the database, there is still a lot of insensitive data in the database. So we proposed the selective tuple encryption method with Bloom Filter. A Bloom Filter [3] is a simple space-efficient randomized data structure used to represent the existence of the data. The filter is a bit array over which membership queries are conducted to distinguish between members of the given set or not. Hash function techniques are used to both save space and allow member lookup. Bloom Filters allow false positives but the space savings often outweigh this drawback when the probability of an error is controlled. The Bloom Filter is an array of m bits, initially all set to 0. The Bloom Filter uses k independent hash functions with range m . The number of the sample space is n elements. After inserting n elements with k hash functions in the m bits Bloom Filter, the probability of the false positive error [3] is $f = 1 - (1 - 1/m)^{kn} \approx 1 - e^{-kn/m}$. This false positive rate could be reduced by choosing the large number of m in proportion to the number of input element n . In Figure 3, The BF means the query conversion process using Bloom Filter value and ED means the selective etuple decryption process. The metadata store the bucket index of the each attribute.

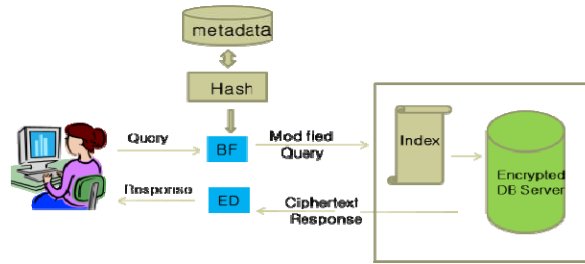


Figure 3. Selective tuple encryption with Bloom Filter method

We propose two kinds of encryption methods for the sensitive data only. The first type is the characteristic data involved in each attribute. The keyword for the search process is extracted from the each characteristic attribute. The city name “seoul” is one example as a keyword. The extracted keyword is hashed and the resulting value of the hash function is considered a bit number and the corresponding bits are set by one in the Bloom Filter, $BF = Hash(seoul)$. The second type is the numerical data involved in each attribute. The numerical data is converted to the bucket index which is stored in the metadata. The grade is one example as a numerical data. The value of the grade would be divided by the large number of the buckets because the bucket index value is not disclosed, and only involved in the Bloom Filter using hash functions. So, the numerical attribute involved in the user query is converted to the bucket index first, then the bucket index value is converted to the corresponding bits of the Bloom Filter, according to the result of the hash functions.

A query originated from a user is transformed to the modified query in order to search the exact data from the encrypted database. We call this method SEBF (Selective Tuple Encryption with Bloom Filter). Table 4 shows the query conversion process in SEBF. The generation process of the encrypted database in SEBF is shown in Figure 4.

Table 4. The query used in SEBF method

The original plain-text query	The transformed query
SELECT name FROM std_info WHERE city = seoul	SELECT s-tuple FROM std_info(E) WHERE BF =Hash(seoul)

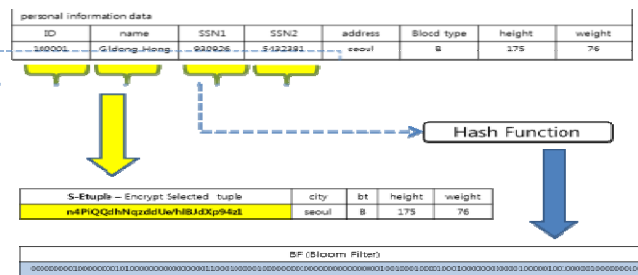


Figure 4. The structure of the SEBF method

4. DISCUSSION AND CONCLSIONS

In this chapter, we will compare the proposed encryption method with the other encryption methods. The First encryption method is the Tuple Encryption with Bucket Indexing method. We call this method as TEBI. This method is proposed by Hore [2, 5, 6] and the same to the typical tuple encryption method. In Figure 5, you can see that it only uses the bucket index regardless of the attribute data type. All of the tuple elements are encrypted by the e-tuple (encrypted tuple) and each attribute of the tuple is converted by bucket indexes. Therefore, we can search the data using the bucket index which is stored in the metadata. The e-tuple data is extracted from the database according to the bucket index. If we want to search the city name “seoul”, then we first convert the city name to the according bucket index using the metadata in a client module. The modified query is sent to the database server, and query results, which are the e-tuples according

Table 5. Two queries used for performance analysis

Query1	Query2
SELECT count(*) FROM std_info, std_garde WHERE (e-grade>750)AND (city =jecheon)	SELECT count(*) FROM std_info, std_garde WHERE (d-id=electronics)AND (grade >average(grade))

Table 6. The result of the search performance test

	Return No (Q1)	Search Time(Q1)	Return No (Q2)	Search Time(Q2)
Plaintext	764	0.0450026	2565	0.117067
TEBI	7388	0.2150123	19981	0.4550263
TEBF	7436	0.2480136	21996	0.618254
SEBF	807	0.0650031	10018	0.310177

Table 7. The comparison between three methods

Encryption Unit	Index	Keyword Search	Numerical Operation	Security	Comment
TEBI	BI	Low	Medium	Low	Disclose the distribution of the data.
TEBF	BI, BF	Medium	Medium	High	High speed only if we use large number of bucket index.
SEBF	BI, BF	High	High	Medium	Only sensitive data should be encrypted

We can see that the SEBF method has the best search performance. But it is a very difficult problem to decide how much data would be encrypted according to the sensitivity. In fact, the search performance is different depending on the queries types we choose. If we use character type cipher-text as a search condition, SEBF has better performance than other methods because it uses keyword search. In the case of the aggregation operation, SEBF has better performance than other methods only if it uses the large numbers of the bucket index.

ACKNOWLEDGEMENTS

This paper was supported by the Semyung University Research Grant of 2015.

REFERENCES

- [1] S. Vimercati, and S. Foresti, "Privacy of Outsourced Data", Auerbach Publications (Taylor and Francis Group), 320: 174-187.
- [2] B. Hore, S. Mehrotra, and H. Hacig .m .s, "Managing and Querying Encrypted data", Handbook of Data Security, pp 163-190, 2008.
- [3] A. Broder and M. Mitzenmacher. Network applications of Bloom Filters: A survey. *Internet Mathematics*, 1(4):pp.485-509, 2004.
- [4] J. Kim, T. Sahama, S. Kim, "A Performance test of Query Operation on Encrypted Database", *LNCS 235*, pp 801-810, 2013
- [5] B. Hore, S. Mehrotra, G. Tsudik, "A privacy-preserving index for range queries", Proceedings of the Thirtieth international conference on Very large databases- Volume30. VLDB '04, VLDB Endowment pp 720-731, 2004.
- [6] H. Hacig .m .s, B. Iyer, C. Li, and S.Mehrotra, "Executing SQL over encrypted data in the database service provider model", In Proc. of the ACM SIGMOD, pp. 45-49, 2002
- [7] D. Shin, T. Shahama, J. Kim and J. Kim, "The Scalability and the Strategy for EMR Database Encryption Techniques", *JICCE*, pp. 556-582, 2011.

BIOGRAPHIES OF AUTHORS

Young-Dal Jang (jang_yd@naver.com) received his B.S and M.S degrees in electronic engineering from Semyung University, Jecheon, Korea in 1998 and 2008. He is currently in his Ph.D course in information and telecommunication in Semyung University, Jecheon, Korea. His current research intersets include the netowrk securityand database security.



Ji-Hong Kim (jhkim@semyung.ac.kr) received his B.S degree in electronic engineering from Hanyang University, Seoul, Korea in 1982. and received his M.S and Ph.D degrees in electronic and communication engineering from Hanyang University, Seoul, Korea in 1984 and 1996. He has worked in the Department of the information and telecommunication in Semyung University since 1991. His current research intersets include the netowrk security and database security and cloud computing.