A Study on the Encryption Model for Numerical Data

Ji-Hong Kim, Tony Sahama, *Member, KIMICS*

***Abstract*—** The encryption method is a well established technology for protecting sensitive data. However, once encrypted, the data can no longer be easily queried. The performance of the database depends on how to encrypt the sensitive data.

In this paper we review the conventional encryption method which can be partially queried and propose the encryption method for numerical data which can be effectively queried. The proposed system includes the design of the service scenario, and metadata.

***Index Terms*—** Bucket, Metadata, Query, Database encryption.

I. INTRODUCTION

Nowadays the organization holds very sensitive information in database. As the volume of this information is increasing rapidly, many organizations decide to outsource their database to external service providers [2]. The main advantage of outsourcing is the reduced cost and more consistent database service providers. As the database is not under the data owner's control, data confidentiality and integrity are important to the outsourced database. In order to provide these functions, the one way to keep sensitive data in the database secure is to encrypt it. Encryption and Decryption must be executed not in the server, but in client modules. The server can looks only the encrypted data and execute queries on encrypted data at.

There are two kinds of database security methods which are termed as an access control method and an encryption method. Access control is used for allowing the only authorized person to access the limited data with the user authentication. But even if it is a secure database system, it can not protect the database system from attacks by an internal user. In addition to that, it can be damaged by the mistake or intention of the database manager.

Encryption is a well established technology for protecting sensitive data. Unfortunately, the existing encryption techniques on database systems cause undesirable performance degradation. There are some encryption methods which are record based encryption, tuple based encryption, and attribute based encryption. Record based encryption is sometimes called block based encryption [3]. It encrypts the bulk of data which include a lot of tuples. But Record based encryption needs a lot of the time to decrypt the bulk of data in order to get the final result data. Tuple based encryption methods [7] encrypt all the data within each tuple. So it encrypts not only the sensitive data but also non sensitive data. Attribute based encryption methods [5] can encrypt the only sensitive data itself, but it needs a lot of padding bits to encrypt the sensitive data. In this paper we review the conventional encryption methods which can be partially queried and propose the encryption method for numerical data which can be effectively queried. In this paper, after a brief explanation of the conventional encryption method for database systems, we focus on the new proposed encryption method.

The remainder of this paper is organized as follows. Section 2 describes the conventional encryption method for database systems and compares their merit and shortcomings. Section 3 presents the new proposed encryption algorithm and Section 4 shows the example for the proposed encryption algorithm. Finally, we conclude with a summary and directions for future work in Section 5.

II. RELATED WORK

There are many studies to execute query on encrypted data. But first, we will describe Bucket

Manuscript received received October 28, 2008; revised March 2, 2009.

Ji-Hong Kim is with the Department of Information and Tele-communication , Semyung University, Jecheon Si, ChounChung Buk Do, 390-700, Korea.(Tel: +82-43-649-1310, Email: jhkim@ semyung.ac.kr) Tony Sahama is with the School of Information Technology, Faculty of Science and Technology, Queensland University of Technology, Brisbane QLD, 4000, Australia. (Email: t.sahama@qut.edu.au)

based, B+ Tree index, Privacy homomorphism, and OPES (Order Preserving Encryption Schema) methods. The first proposal toward the solution of this problem was the bucket based indexing method using a number of buckets on the attribute domain [7]. Bucket based indexing methods use to partition attribute value range in a number of non overlapping subsets of values, called buckets, containing contiguous values. The other major proposal was the B+ Tree index method presented [4] where authors proposed storing additional indexing information together with the encrypted database. Such index information can be used to search the right information from the encrypted data in the database. The B+ tree structure is typically used inside the database. Another method is using Privacy homomorphism. It has also been proposed for allowing the execution of aggregation queries over the encrypted data in database system [5]. Therefore, the operation on an aggregation attribute can be evaluated by computing the aggregation at the server site and by decrypting the result at the client side. This technique causes the aggregation operation to be executed on the server side instead of the client side. Most of papers about privacy homomorphism focus not on comparison operations but on arithmetic operations. An OPES is presented to support equality and range queries over the encrypted data [8]. Because the encrypted data has preserved order, equality, range query can be operated on the encrypted data in the database. This approach operates only on integer values and does not operate on non integer values.

To summarize, Table 1 [1] shows, for each indexing method discussed, what type of query is supported. A hyphen(-) means that the query is not supported, a circle(O) means that the query is supported and a (P) means that the query is partially supported.

Table 1 Supporting queries for Indexing methods

Index	Equality Query	Rang Query	Aggregation Query
Bucket- Based	O	P	-
B+ Tree	O	O	O
Privacy Homomorphism	O	-	O
OPES	O	O	P

III. THE PROPOSED ENCRYPTION METHOD

3-1 The design on the encrypted database system.

We use the following Database service scenario presented in Fig. 1 below. Basically it is client/server module. The client module has extra metadata and encrypt/decrypt modules.

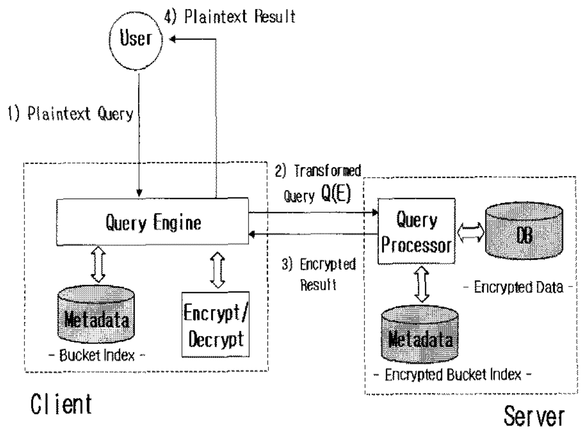


Fig. 1 : The Encrypted Database Service Scenario.

The encrypted database service scenario in Fig. 1 involves the following entities and components.

1. The user (human entity) requests the queries to the system
2. The client (front end) transforms the queries posed by user(s) into equivalent queries operating on the encrypted data stored on the server
3. The server (Database) stores the encrypted data from one or more data owners and makes them available for distribution to clients
4. The Query Processor in the client module transforms the original plaintext query to the transformed query according to the used cryptographic algorithm
5. The Query Executor in the server process the transformed query and return the encrypted query result to clients

Metadata is used to help search data in database. In the encrypted database system, metadata would be the index to help search the result.

First of all, data owners encrypt the sensitive data and create metadata in order to facilitate the search of the encrypted database. This metadata is stored in the client side. Then, the user sends the database query with plaintext form as usual.

The Query engine in the client module transforms Q to the transformed Q(E) with the help of the metadata, which is constructed by the data owner beforehand, in order to search data in the encrypted database and send the transformed Q(E) to the server. So, the proper construction of metadata is very important.

In the database server, Query Processor queries the encrypted database server about the result. The database server gets the result and sends it to client.

Finally, the client returns the exact encrypted result to the client and Query engine in client module extracts, that was, exact result with the help of the metadata.

3-2. Example

Here we use employee databases which consist of Emp ID, Name, Year, Dep., and Salary fields like Table 2. We will encrypt the salary fields because it is sensitive data. We use bucket index method per bucket in order to query the database.

Table 2 An example of the sample database.

Emp ID	Name	Year	Dep.	Salary
1	Ann	1980	Eng-1	25,000
2	Bob	1975	Sale-2	30,000
3	Carol	1970	Sale-1	35,000
4	David	1975	Eng-2	32,000
5	Ann	1970	Dev-1	34,000
6	John	1985	Dev-2	13,000
7	Merry	1977	Dev-1	30,000
8	Tom	1965	Sale-1	35,000

First of all, we divide the salary domain into several buckets like Table 3 and assign the Salary_index for bucket 3 in Table 4.

Table 3 Bucketing of salary data.

Salary	[10,000, 20,000)	[20,000, 30,000)	[30,000, 40,000)	[40,000, 50,000]
Bucket ID	5	2	3	6

Table 4 Salary_index for Bucket 3.

Salary	30,000	32,000	34,000	35,000
Salary_index	(3, 1)	(3, 2)	(3, 3)	(3, 4)

Table 5 The encrypted Salary Data in Bucket 3 (Bucket Info : BK, IV)

Emp ID	Name	Year	Dep.	Salary(E)	Salary Index
2	Bob	1975	Sale-2	C1 = CBC(IV,M1,BK)	(3, 1)
3	Carol	1970	Sale-1	C4 = CBC(C3,M4,BK)	(3, 4)
4	David	1975	Eng-2	C2 = CBC(C1,M2,BK)	(3, 2)
5	Ann	1970	Dev-1	C3 = CBC(C2,M3,BK)	(3, 3)
7	Merry	1977	Dev-1	C1 = CBC(IV,M1,BK)	(3, 1)
8	Tom	1965	Sale-1	C4 = CBC(C3,M4,BK)	(3, 4)

The data stored in metadata storage about bucket 3 are Bucket Info(Bucket Key, Initial Value) and Salary_index (Bucket ID, SN). The initial Value is a boundary value that defined the starting point of bucket entry. In this example, the Initial Value of Bucket 3 is 30,000. Bucket Key is the key used to encrypt the corresponding entry and SN(Sequence Number) means the ciphertext equivalent to the corresponding plaintext. In this example, these are (1 : 30,000 – C1), (2 : 32,000 – C2), (3 : 34,000 – C3), (4 : 35,000 – C5).

We use the CBC (Cipher Block Chaining) mode algorithm to encrypt the sensitive data. The CBC method is very effective to encrypt sequential blocks continuously. CBC(IV,M1,BK) means that plaintext M1 is encrypted using CBC algorithm with IV(Initial Vector) and BK(Bucket Key).

Table 6 The data stored in bucket 3 metadata.

Salary_index	Salary
(3, 1)	M1 = 30,000
(3, 2)	M2 = 32,000
(3, 3)	M3 = 34,000
(3, 4)	M4 = 35,000

The (3, 1) in Salary_index means that the 1st Salary data in bucket 3 is M1,

3-3. An analysis of queries.

In this section, we use three query examples to illustrate the proposed mechanism more clearly.

User's query Q is reconstructed with Q(E) with the help of metadata and encrypt modules in client.

(ex 1) Equality Query

SELECT name
FROM employee
WHERE Salary = 35,000

Client module receives the above plaintext query. Then, client module inquires to metadata about the key material in which salary = 35,000 is included.

Metadata will return Bucket Info (BK, IV) and Salary_index (bucket ID, SN). In this example, bucket ID =3, Initial value IV, bucket key = BK, and SN=4.

Query Engine in client modules can encrypt the original Equality Query like this.

SELECT name
FROM employee
WHERE Salary(E) = CBC(C3,M4,BK)

In the encrypted Equality Query, the encrypted Salary is C4= CBC(C3,M4,BK). C4 can be calculated by C3, C3 by C2, C2 by C1.

That is, the data stored in bucket 3 metadata is used to calculate C1, C2, and C3 successively

C1 = CBC(IV,M1,BK), C2 = CBC(C1,M2,BK), C3 = CBC(C2,M3,BK)

(ex 2) Range Query

SELECT name
FROM employee
WHERE Salary < 31,000 AND Year > 1974

First of all, "Salary < 31,000" consists of bucket ID = 5, 2, and 3 partially. The range of bucket ID 3 is [30,000, 40,000). So, according to the Salary_ index in Table 4, "Salary <31,000" means that it's Salary_ index is (3, 1).

Query Engine in client modules can encrypt the original Range Query like this.

SELECT name
FROM employee
WHERE (Salary^{id} = 5 AND Year > 1974) OR (Salary^{id} = 2 AND Year > 1974) OR (Salary^{id} = 3 AND Salary_index = (3, 1) AND (Year > 1974))

Table 7. The table satisfying "Salary <31,000 and Year >1974"

Emp ID	Name	Year	Dep.	Salary	Salary_ Index
1	Ann	1980	Eng-1	25,000	(2, 1)
2	Bob	1975	Sale-2	30,000	(3, 1)
6	John	1985	Dev-2	13,000	(5, 1)
7	Merry	1977	Dev-1	30,000	(3, 1)

Then the server returns the encrypted Salary and name and Salary_index fields. Then client can then make direct calculation using the relevant bucket Info and Salary_index stored in metadata.

(ex 3) Aggregation Operation

SELECT AVR (Salary)
FROM employee
WHERE (Dept. = Dev-1) AND (Dept. = Dev-2)

First of all, we find the tuples whose employee is in Dev-1 and Dev-2. Then, find the sum of the salary.

Table 8. The table satisfying the condition

Emp ID	Name	Year	Dep.	Salary(E)	Salary _index
5	Ann	1970	Dev-1	C3 = CBC(C2,M3,BK3)	(3, 3)
6	John	1985	Dev-2	C1 = CBC(IV,M1,BK5)	(5, 1)
7	Merry	1977	Dev-1	C1 = CBC(IV,M1,BK3)	(3, 1)

In this case, a plaintext query is sent to the server directly, and then the server returns the encrypted Salary and dep. and Salary_index fields. Then client can then perform directly calculation using the relevant bucket Info and Salary_index stored in metadata.

Finally, you can know that it is very easy to get the result in case we want to get the sum or average of salary about Salary range, for example, 20,000 < Salary < 30,000. Then the client could get the exact result using the successive decryption of the encrypted Salary data in bucket 2.

IV. CONCLUSIONS

Bucket method is generally fast to query the encrypted data, so we use the bucket method. If we use B+ tree index method instead of the bucket method, then the index file will be very large as the amount of database is getting increased. This means that using the suitable number of buckets and metadata is very important to determine system performance.

In our suggestion described in previous sections, metadata plays a very important role as the simplified index on the encrypted data. Bucket Info, which are it's IV and Key, must be kept secure in client module. Metadata on each bucket may be stored in both client and server. But if it is stored in the server, bucket_index and plaintext sequence should be encrypted and stored in the server module.

Secondly, the CBC mode that we used is the encryption algorithm that makes the speed of encryption fast because it encrypts each item of the bucket sequentially. So it takes one round of processing time to encrypt /decrypt all of the Salary fields in one bucket. If we use the ECB (Electronic Codebook) mode, then it takes more time than using the CBC mode because it encrypt /decrypt each Salary field respectively

Thirdly, we can compose bucket flexibly. Now we composed bucket in terms of the Salary range. But if another criteria of bucket composition is more important and frequently used, then we can compose the bucket with respect to another parameter. For example, if they need the salary data with respect to a particular department in our example, then we can construct the bucket composition to suit to that department.

Finally, the mechanism is easy to delete and insert the new item. If we want to insert the new record, then insert the new record, re-encrypt the bucket successively, and modify the metadata information about the corresponding bucket.

It is possible for the client module to encrypt and decrypt data with the help of the metadata within the client module. Metadata in the client module plays a very important role in processing the searching query for the encrypted data in database.

In this paper we have designed the practical database model and metadata schema that is suitable to the encrypted data in the database system. This proposed system will serve the good model for the protection of the sensitive data in database system.

What kind of encryption algorithm is used is out of the scope. But we will study the more effective encryption algorithm applicable to our model and analyze the performance about the proposed model in detail and propose better suited models to the real database system

.

ACKNOWLEDGEMENT

This paper was supported by the Semyung University Research Grant of 2008.

REFERENCES

[1] Sabrina De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, P. Samarati, "Privacy of Outsourced Data," in Digital Privacy: Theory, Technologies and Practices, Auerbach Publications (Taylor and Francis Group),

[2] Hakan HacigÜmÜs, Bala Iyer, and Shrad Mehrotra, Providing database as a service. In Proc, of the 18th International Conference on Data Engineering, San Jose, California, USA, IEEE Computer Society, 2002, 29.

[3] Bala Iyer, Shrad Mehrotra, Einar Mykletun, Gene Tsudik, and Yonghua Wu, A Framework for efficient storage security in RDBMS. In Bertino, E. et al., Eds., Proc. of the International Conference on Extending Database Technology(EDBT2004), v-2992 of Lecture Notes in Computer Science, Crete, Greece. Springer, 2004, 147.

[4] Ernesto Damiani, S.De Capitani di Vimercati, Sushi Jajordia, Balancing confidentiality and efficiency in untrusted relational DBMSs. In Jajodia, S., Atluri, V., Eds., Proc. of the 10th ACM Conference on Computer and Communications Security(CCS03), Washington, DC, USA, ACM, 2003, 93.

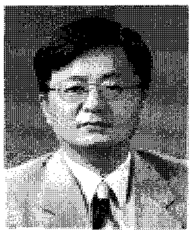
[5] Hakan HacigÜmÜs, Bala Iyer, and Shrad Mehrotra, Efficient execution of aggregation queries over encrypted relational databases, In Lee, J., li, J. Wudhang, K., and Lee, D., Eds., Proc. of the 9th International Conference on Database Systems for Advanced Applications, Volume 2973 of Lecture Notes in Computer Science, Jeju Island, Korea, Springer, 2004, 125.

[6] G. Aggarwal, et al. Two can keep a secret: a distributed architecture for secure database services, In Proc. of the Second Biennal Conference on Innovative Data Systems Research(CIDR 2005), Asilomar, CA, 2005, 186.

[7] Hakan HacigÜmÜs, Bala Iyer, Chen Li, and Shrad Mehrotra, Executing SQL over encrypted data in the database service provider model. In Proc. of the ACM SIGMOD 2002, Madison, Wisconsin, USA. ACM Press, 2002, 216

[8] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, Yirong Xu, Order preserving Encryption for numerical Data, in Weikum, G., Konig, A., and Debloch,s., Eds., Proc. of the ACM SIGMOD 2004, Paris, France. ACM, 2004, 563.

[9] J. Domingo I Ferror. and J. Herrera-Joancomarti. A privacy homomorphism allowing field operations on encrypted data. I Jornades de Matematica Discreta I Algorismica, Universitat de Catalunya, March 1998.



Ji-Hong Kim
Is a professor at the Department of Information Tele-communication, Semyung University. He received the M.S. and Ph.D. degrees in Electronic Communication engineering from Hanyang University in 1984 and 1996, respectively. His interests include PKI, Database security, and cryptographic applications.

Tony Sahama
Is a Senior Lecturer at the School of Information Technology, Faculty of Science and Technology, Queensland University of Technology. He received the PhD degree in Computer Science from Victoria University, Melbourne in 1999. His interest includes Computer Experiments, Modeling and simulation, Medical Informatics and Information Technology application in Health care.