A Technique for Handling the SQL Aggregate Functions over Encrypted Data

Shaukat Ali[1] Azhar Rauf[2] Huma Jave[3]

GJCST Classification
E.3,H.2.3

***Abstract*-Data is a valuable source for the development of any organization. Most of the data are stored in the databases which necessitates its security. Many techniques can be used for database security among them one is the encryption of data in tables. Encryption is an effective security layer however, it has a flaw of performance degradation. Many techniques have been suggested in the literature to improve the data retrieval process in the encrypted data column, but some problems are unresolved, for example, the aggregate functions' query. This paper proposes a new technique to run the aggregate function of SQL directly on encrypted data. The experimental results prove the hypothesis.**

Keywords-Database security, Encryption, Performance.

I. INTRODUCTION

Databases contain important data which is necessary for the improvement and running of organization. Many layers have been used for the security of data in the databases, for example, identification, authorization [1], view [2], MAC [3], DAC [4]. In many cases these traditional security layers can provide security, but in case of more security requirement encryption is considered as a favorite option. In the case when an intruder penetrates all the security layers and is able to retrieve data, encryptionmakes sure, the hacker cannot use important data. Encryption is the final layer of database security [5]. It secures data even after it has been stolen or hacked by intruder. Encryption is affective but is also has problems, for example, SQL (Structure Query Language) queries cannot be run directly on it. SQL is one of the core services of the DBMS (Database Management System). Whenever any operation is performed on the encrypted data, first the encrypted data needs to be converted into plain text and then the SQL operation can be performed on it which is expensive performance wise. Encryption has a disadvantage of performance degradation. Many algorithms have been proposed to improve the performance, but still problems persist. One of these problems is to handle the aggregate functions of SQL. Agrawal et al [6] proposed a new encryption technique for numerical values. They used order preserving technique to handle the problem of range queries plus some of the aggregate functions. This new technique can handle MAX, MIN, and COUNT type aggregate function's query and cannot handle SUM and AVG aggregate type queries. It is a useful encryption technique for numerical data in which the SQL queries can be run directly on it, but it has a problem that it can not be generalized to the all encryption algorithms. The order persevering technique does not provide adequate security [7]. This research proposes a new technique for handling the aggregate function queries in the encrypted data column. The proposed technique uses two tables for a single table's data. One is the actual table and other, a dummy table, is used for the aggregate function's query only. When a query comes to database, so it nature will be checked if it has an aggregate function's query on encrypted column, then it will transformed to dummy table to retrieve the aggregate function value from dummy table and the rest of values (unencrypted) will be retrieve from actual table.

Rest of the paper is organized as follow:

Section II is about the related work done in the area. Section III is our research statement and hypothesis for that research problem. Section IV is the proposed method and its working methodology. Section V is flow chart and section IV gives the algorithmic steps for the proposed technique. Section IIV shows the experimental results and section IIIV concludes the work done.

II. RELATED WORK

The most relevant work to run the aggregate functions directly on encrypted data is done by Agrawal et al [6]. They proposes and order preserving encryption technique which support the range queries and some of the aggregate functions for example MIN, MAX, and COUNT, but it can not work on the SUM and AVG aggregate functions. Sesay et al [8] proposes a new encryption scheme that eliminates the decryption of total data for querying, but it fails to work in the case of aggregate function type queries including SUM and AVG aggregate function. According to Sion [9] handling aggregate functions in the encrypted data is an open area of research. Literature [10, 11] uses hashing index method search in the encrypted data column. This method is useful for improving the efficiency of SELECT query while searching in encrypted data column, but these schemes do not work for the queries having aggregate function. Bucketization is another technique used to speed up the performance of query while searching in the encrypted data column. This scheme improves the performance by dividing the searching domain into buckets. Literature [12, 13] uses bucketization technique, but these techniques do not support the aggregate functions type queries. The literature survey reveals that aggregate function's queries, over encrypted data column is a problem.

About-*Department of Computer Science, University of Peshawar, KhyberPukhtoonkhwa ,Pakistan
(e-mail-shaukat191@yahoo.com[1], azhar.rauf@upesh.edu.pk[2]
humajaved15@upesh.edu.pk[3])

III. RESEARCH PROBLEM AND HYPOTHESIS

There is no direct method to run aggregate function of SQL on the encrypted data. Hypothesis of the proposed solution is to keep sensitive data column in unencrypted form in another table. This will resolve the above mentioned problem.

IV. PROPOSED METHOD AND ITS WORKING METHODOLOGY

This paper proposes a new method for directly running the aggregate function of SQL on encrypted data column. In this method two tables are used for that table which have encrypted data column. One table is the actual table and other one is just a dummy table. The dummy table will contain the encrypted data column of the actual table in the unencrypted form and hash values of that column which is used in the WHERE clause. The order of the rows will be shuffled in the dummy table in order to provide security even it is accessed by unauthorized person. There is no direct link between the actual table and this dummy table. The dummy table will be stored in secured schema. The dummy table will only be used for the aggregate function's queries.

The working methodology will be follows

When user performing query to the table having encrypted data column, so its nature will be checked. If the query having aggregate function on the encrypted data then it will be transformed to the dummy table to solve the aggregate function. The dummy table can only be accessed by thoseusers who have clearance to the encrypted data. In this method there is no need to decrypt any value.

Table 4.1: Actual_Table

Name	Salary	Job Title	Company Name
Ikram	Encrypted	Manager	Stop-Loss 200
Umar	Encrypted	Assist manager	Atlanta Medical Services
Shahid	Encrypted	N/A admin	First Midwest Financial

TABLE 4.2 Dummy_table

XYZ (Salary column of Actual_Table)	Hash Values of company name
12000	41789000916342
10000	41789000916346
9000	14146267157396
13000	41789040916342

EXAMPLE

Reference to Table 4.1, consider the user's following query over the Actual_Table.

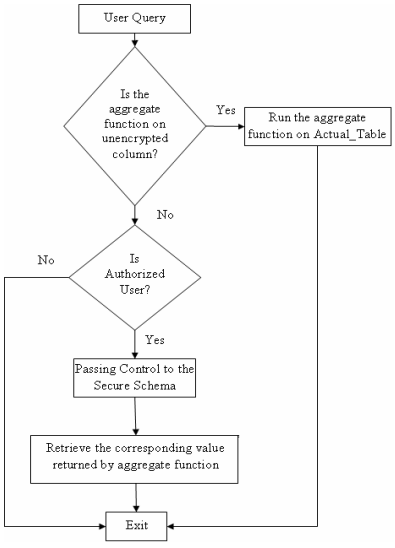
```
SELECT Emp_Name, SUM(Salary)
FROM Actual_Table
WHERE [Company Name] = _Stop-Loss 200'
GROUP BY Emp_Name
```

The algorithm interprets the above query and transform as follow:

```
SELECT Emp_Name, (SELECT SUM(Salary) FROM
Dummy_table WHERE [Hash Values of company name]
=HashvalueGerateFunction(Stop-Loss 200))
FROM Actual_Table
GROUP BY Emp_Name
```

Here, in the first query if user wants to find sum of the salaries of all employees in a particular company, so he/she must decrypt that values and then retrieves the required aggregated value of sum function. In case of proposed approach the query is intercepted into the form as shown in example. In the case of second query there is no need to decrypt the values for finding the sum of salaries of employees. The inner query calculates the required aggregate function value directly from the dummy table which increases system performance.

V. FLOW CHART OF THE PROPOSED SYSTEM3



V. ALGORITHM

Following are the algorithmic steps of the proposed algorithm

1. [User Query]
   User poses query
2. [Check the aggregation Column]
   If (aggregation function is not on encrypted column)
   Goto step 3
   Else if (Authorized User)
   Goto step 4
   Else
   Goto step 5
3. [Perform aggregate function on Data]
   Run the aggregate function on Actual_Table
   Goto step 5
4. [Passing Control to the Secure Schema]
   [Run aggregate function on column in secure schema]
   Retrieve the corresponding value returned by aggregate function
5. Exit

VI. SECURITY IN THE PROPOSED SYSTEM

Sensitive data column in database can be categorized in to the following two types:
i. Independent data column
ii. Dependent data column

A. Independent data column

The data column which can be used independently for some information leakness for example the prepaid cards number of a telecom company. If the prepaid card number is stolen by someone so he/she will simply use it without any extra information.

B. Dependent data column

The dependent data columns are those whose information can not be used independently. Its data can be informative when the data of some other column is combined with it. For example salary in the employee table of anorganizational database. Salary information is sensitivewhen the employee record is available and if the employee
record does not exist then the salary information ismeaningless. similarly passwords column which needs user
name to get information from it.An organization has a lot of data in which some is verysensitive and important for that organization. All of the datain organization may not be as much sensitive but some maybe sensitive. Similarly in a table all the columns may not beimportant for the organization security point of view. Inproposed technique security is proposed at column level asall the columns may not be sensitive. Sensitive column is encrypted in the actual table and a copy of the same columnis stored in another dummy table which only used for the aggregate type queries on the encrypted column. First security layer is that the dummy table is stored in a secure schema. Only those users will have access to secure schema that have clearness to the encrypted data. The proposedsystem is very secure in the case of Dependent data columnas there is no direct relationship between the actual table anddummy table. If the secure schema layer is broken by hackerstill no information can be extracted from data in the case of
dependent data.

VII. EXPERIMENTAL RESULTS

Proposed algorithm has been tested on TPC (Transaction Processing Performance Council) [14] schema and data. A table of TPC-E schema named Cash_transaction is used for testing purposes. It has total 133152 records. Different amount of data has been retrieved for different queries having aggregate function.

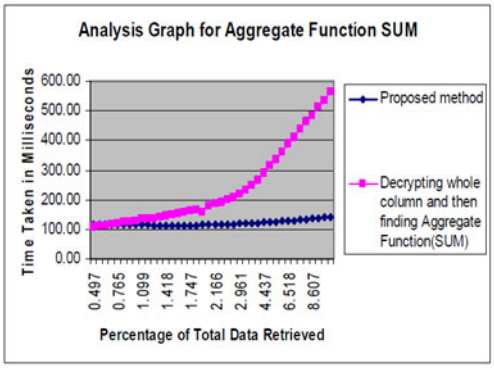


Figure 7.1: analysis graph of SUM aggregate function

The figure 7.1 shows the analysis graph of SUM aggregate function. It is shown in graph that if the queried rows are less than the 0.7 % of total rows, retrieved from table then the state of art technique is better than ours, but greater than 0.7 % gives efficient results in the proposed technique.

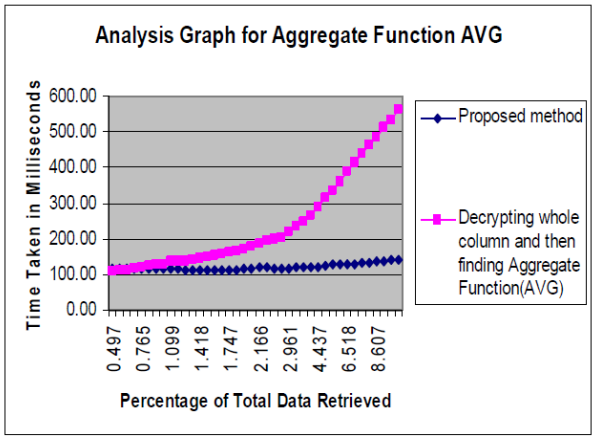


Figure 7.2: Analysis graph of average (AVG) aggregate function

The graph (Figure 7.2) is the experimental results of aggregate function AVG (average). Here, again the results of the state of art technique is better in the case of less data retrieval upto 0.7 % of total rows but expensive performance wise when selected rows in the SELECT query is more than the 0.7 % of total rows of table.

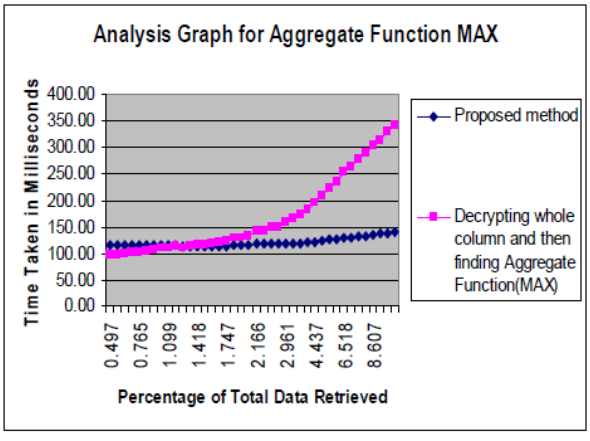


Figure 7.3: Analysis graph of MAX (maximum) aggregate

Function

The graph (figure 7.3) is the experimental results of the aggregate function MAX‘s query. It is obvious from the graph that the proposed system is better when we find the maximum value in more than the 1.5% of total rows in table.

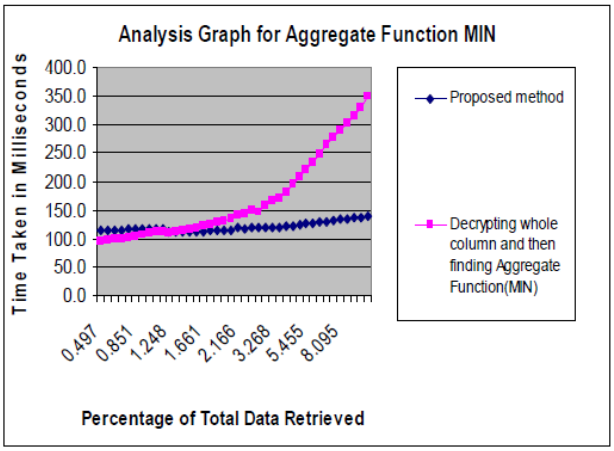


Figure 7.4: Analysis graph of MIN (minimum) aggregate Function

Figure 7.4 shows the graph of experimental results of aggregate function MIN (minimum) performance forproposed and state of arts technique. Again like the MAX, the MIN aggregate function is also better in the case of rows affected by query is more than 1.5 % of total rows in the table.

If we look to all the four graphs so the results is better in the case of proposed technique over the state of art technique. We used that state of art technique in which the total effected rows are decrypted and then the aggregate function is run on it. All the results for the proposed technique are better when the total effected rows of a table are more than 1% of total rows in that table. In the typical environment the aggregate function is run on the data in which at least more than 1% rows are selected. For example in a company there are ten different groups of worker and manager of thecompany is interested to find the minimum, maximum,average, or sum of salaries of all workers in a particulargroup then the probability of rows to be affected, in the table

of workers, is 10% of total rows. It means that in the typical environment the rows affection is more than 1% of the total.

VIII. CONCLUSION

This work proposes an efficient technique for handling aggregate function‘s query over encrypted data. All tests conducted on the TPC schema and data. Results of the experiments are satisfactory. The proposed technique is efficient when the rows affection in the SELECT query having an aggregate function is more than 1% of the total rows.

IX. REFERENCES

1) Ambhore, P.B. Meshram, B.B. and Waghmare, V.B. (2007). A implementation of object oriented database security, in Fifth International Conference on Software Engineering Research, Management and Applications. 359 – 365.
2) Bertino, E. Betini, C. Ferrari, E. Samarati, P.(1996). Supporting periodic authorization andtemporal reasoning in data base access control, in 22nd Intl. Conf. On very Large Data bases.Bombay (India). 472-483.
3) Bertino, E. Samarti, P. Jajodia, S. (1997). An extended authorization mode, in IEEE Trans. On knowledge and data Engineering. 85-101.
4) Sloan, J.A., and Sloan, R.H. (2004). A layered design of discretionary access controls with decidable properties, in IEEE Symp.Security and Privacy. 56–67.
5) Kiely, D., (2006). Protect Sensitive Data Using Encryption in SQL Server 2005..
6) Agrawal R., Kiernan, J. Srikant, R. Xu,Y. (2004). Order Preserving Encryption for Numeric Data, in SIGMOD 2004. ACM New York, NY, USA Paris, France. 563 - 574.
7) Huang, Q., (2009). Research on ciphertext index method for relational database, in 2009 2nd IEEE International Conference on Computer Science and Information Technology. Beijing, China. 445-449.
8) Sesay, Z.Y., Chen, J. and Xu, D. (2004). A Secure Database Encryption Scheme, IEEE. 49-53.
9) Sion, R., (2005). Query Execution Assurance for Outsourced Databases, in 31st VLDB Conference. Trondheim, Norway. 601-612.
10) Wang, Z. Wang, W. Shi, B.L. (2005). Storage and Query over Encrypted Character and Numerical Data in Database in Computer and Information Technology. 77-81.
11) Zhang, Y. Wang.-X.L., Niu, Z.M. (2008). A Secure Cipher Index over Encrypted Character Data in Database, in Seventh International Conference on Machine Learning and Cybernetics. IEEE: Kunming. 1111-1116.
12) Hacigumus, H. Iyer, B. Li, C. and Mehrotra, S. (2002). Executing SQL over encrypted data in the database-service-provider model, in SIGMOD02. ACM. 216–227.
13) Tang, Y. and Zhang, L. (2005). Adaptive Bucket Formation in Encrypted Databases, in IEEE International Conference on e-Technology, e-Commerce and e-Service on e-Technology, e-Commerce and e-Service. IEEE Computer Society Washington, DC, USA. 116 - 119.
14) TPC Benchmark Specification, http://www.tpc.org/