An Effective Data Warehouse Security Framework

Vishnu B
Department of Information Science and Engineering
Acharya Institute of Technology
Bangalore, India

Manjunath T N
Department of Information Science and Engineering
Acharya Institute of Technology
Bangalore, India

Hamsa C
Department of Information Science and Engineering
Acharya Institute of Technology
Bangalore, India

ABSTRACT
In today's electronic world, everyday data is being generated every minute, for every online transaction, in all domains. Since data being very important for any business domains, masking data is a very important entity. Now in most of the enterprises all of these data are stored in a Data warehouse. Since Data warehouse stores the sensitive data to any business enterprises, this is a common target for the hackers to leak the data. Therefore providing security to these Data warehouses is a challenging task. Now we propose a Data masking technique that will protect these Data warehouses. Data masking is a technique in which we replace the original set of data with another set of data that is not real but realistic. The numeric data is masked using a mathematical formula that makes use of modulus operator. Here we also make use of injecting false rows that increases the overall data security strength by creating randomness in the data. Implementation of this method is done on a real-world data warehouse and is implemented on oracle 10g and the results show that the method is a better onethan the existing solution.

Keywords
Data warehouse, Data masking, Encryption, Data Security.

1. INTRODUCTION
Data warehouses store huge amount of data which is highly sensitive for the business evaluation. [7] Hackers try to leak the sensitive information from the data warehouse. This results in business impact in business market. Now efficiently securing the Data Warehouse is very critical [6] with the increase in its complexity and the number of attackers.

1.1 Existing methodology
For the protection of the Datawarehouses the common methods used are Swapping, Substitution, Number and Date variance, Encryption, Nulling out, etc. [2, 4, 10] but among these the most common one used is Encryption. [2] But all the above techniques have some key costs involved such as:

(a) Cleaning up Data: When there is a need to change the data we need to first unmask the data to get back the original data. This is a very tedious work. This condition arises when we use substitution method.
(b) Extra Storage Space: Since Data Warehouse contains huge sets of data even a small modification in the data may introduce large amount of storage space.
(c) Field Overflow: On replacing the real data with false data there is a possibility that the false data will overflow the previously allocated storage capacity.
(d) Query Response Time Overheads: Since there are huge amount of data the response time for processing the queries is usually more in this case.

Since a Data Warehouse itself takes up a lot of storage space using the current encrypting techniques, [8] this proves to be a really costly affair to use it, as they increase the time overheads as well as the storage space. [12] It is also known that increase in the query response time overheads degrades the database performance. Thus the current data masking and encryption techniques are unsuitable for Data warehouses.

1.2 Proposed Method
We propose a solution that helps in minimizing the drawbacks that occur in the existing techniques used to its maximum extent. Unlike techniques where the security strength of the data is more but decrease in the performance or increase in the storage space we try to bring in a balance between the data security strength as well as the database performance. In our proposed method, we do not store the original data anywhere. Initially the DBA enters the data onto the database and once all the data entry is over the DBA specifies which all columns has to be masked and they are masked instantly .Therefore at any instance of time from then only the masked value is stored in the database which wards off any attacks instantly. So when a user requests a query it is first rewritten automatically by a broker which is in between the user application and the DBMS and sends it to the DBMS to process it correctly. Now to mask the numeric data in the database we make use of a mathematical formula that has MOD (modulus) operator (which returns the remainder) and a set of other basic arithmetic operators. Usually most of the data in business enterprises are numeric data which are called as facts. A Fact table having numeric data is used for business evaluation purpose. We also inject false rows onto our database which uses up extra space but helps in increasing the randomness in the database that misleads the hackers.

2. RELATED WORK
An extensive survey on the different Data masking techniques is done in [2]. There is also a Data masking pack that is introduced by oracle [3, 19] which allows us to choose between many of the different techniques that can mask data. Another research work also talks about the various Data masking techniques [4]. It also gives a detailed algorithm of the random replacement technique and also preserving the integrity of the masked data. A survey on the recent trends in Data masking techniques was done in [5]. It gives a detailed description of the Data masking architectures along with the recent development in the field of Data masking.

3. FUNCTIONAL ARCHITECTURE
The Functional architecture has 4 components namely:

i) User/Client Applications
ii) E-MOBAT
iii) BlackBox
iv) Database and its DBMS.

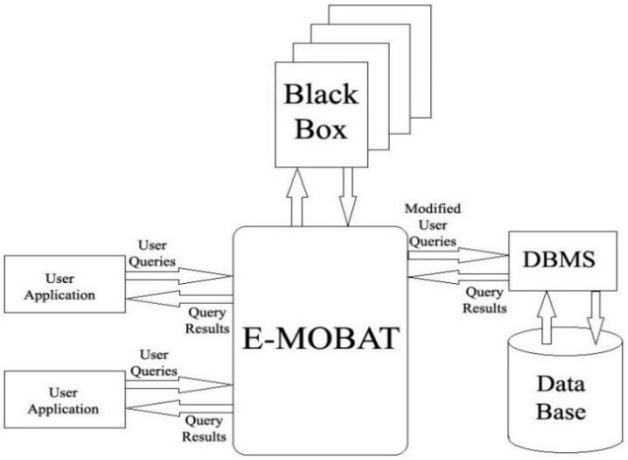

Fig 1: Functional Architecture of Data warehouse security framework

3.1 User/Client Applications

This is also called as the front end. This is usually a Graphical User Interface (GUI) that displays various options for the user to post their request. The requests are of two types- Request by the Administrator (usually for managing the database) and the Request by the user (usually request for some transactions).These requests are sent to the E-MOBAT in the form of queries.

3.2 E-MOBAT

It acts as an interface/broker between the User/Client Application and the DBMS. Its main task is to rewrite the queries sent by the User/Client application using the masking formulae. All necessary masking keys needed are taken from the Blackbox. The E-MOBAT also contains a history log that stores the auditing and control actions. The User or the Administrator performs actions in a particular pattern. The patterns are also pre-stored in the history log. If there is any access to the E-MOBAT in any other pattern, it automatically sends an alert the administrator.

3.3 BlackBox

A BlackBox is a set of files that are created for each masked database. The Blackbox can be accessed only by the E-MOBAT. It stores the masking keys that we use in our masking formulae and the data access policies of database. All BlackBox contents are encrypted using AES standard encrypting algorithms. The BlackBox once lost cannot be restored. The masking keys need to be cracked.

Another feature of the BlackBox is that it automatically shuffles the masking keys after certain duration of time. This is decided by the administrator.

3.4 DBMS and the masked database

The DBMS is the system that manages the data transactions. Masked database contains the data which is in its masked form. The DBMS receives the modified queries from the E-MOBAT, processes it and retrieves the required data and sends it back to the E-MOBAT.

4. IMPLEMENTATION METHOD

A DBA will have the rights to mask data in the database. The DBA enters the login ID and password and try to login to the database. If it succeeds, it will create the BlackBox and stores the data access policies for the users and the permissions. An action log is also created for storing all the further user actions. Now the E-MOBAT asks the DBA which all columns need to be masked. All the necessary masking keys will be generated, encrypted and stored in the BlackBox. Finally the Data masking formula is applied onto the data that has to be masked. Whenever we have to update any data it should be done through the E-MOBAT. Now this will apply the formula to demask the data and the updates can be done.

5. THE DATA MASKING TECHNIQUE

We are keen on masking the numeric data in the data warehouse. This technique that we use has reduced the query response time overhead as much as possible which in turn improves the performance. To understand the technique in more detail, consider a table 'T' with a set of 'm' rows and 'n' columns, where rows are given by (R_1, R_2,……,R_m) and columns are given by (C_1, C_2,……,C_n). Let the value that has to be masked be represented as a pair of (R_i, Cj) where 'R_i' is the row and 'C_j' is the column in which the data that has to be masked. In order to mask the data we must to have three masking keys-

- K_1, a 128-bit random number generated which remains constant for the table T.
- K_2, a 128-bit random number generated which remains constant for a particular column C_j. It is represented as $K_{2,j.}$
- K_3, a number between 1 and 2^{128} that remains constant for a row. The value depends on the data of that particular row. It is represented as $K_{3,i}$.

Here $K_1$and K_2 (called as the private keys) are stored in the BlackBox in their encrypted form. K_3(called as the public key)is stored in the masked table along with the other data. If we store them in another table like a lookup table then we need to make use of table joins which is a very time consuming process, hence it is avoided. Every row of a table will compulsorily have a public key K_3. Consider a value to be masked represented as (R_i,C_j). The new masked value (R_i,C_j)` is given by the formula-

$$R_iC_j = (R_iC_j) - \left((K_{3,i}\%K_1)\%K_{2,j}\right) + K_{2,j} \ldots\ldots \quad (1)$$

Now to retrieve the original value (during the updating of the value by the DBA) the formula used is-

$$R_iC_j = (R_iC_j) + \left((K_{3,i}\%K_1)\%K_{2,j}\right) - K_{2,j} \ldots\ldots \quad (2)$$

Now while inserting the public key $K_{3,i}$ we can do it in two ways- Either by adding a new column to the existing table or by recreating the table along with an extra column that stores the value of $K_{3,j}$. Though in the second option it has more effort and time required for its implementation it is better to go with it since the response time overheads in the case of recreating the table along with the public key is better. Another option is to take a long integer typed column from the table which is already part of the original structure of the table and insert the public key $K_{3,j}$ into that which reduces storage space overheads. To increase the overall security strength once the necessary data has been masked we inject false rows onto the database. Now as we said before that every row has a public key K_3,we can use it to identify which row in the database is a valid row and which one is not. Instead of generating some random value for $K_{3,I}$ for a valid row we have the value of $K_{3,i}$ equal to a multiple of the sum of all values of that row. Then we proceed with the injection of the false rows. Therefore any row with its value of $K_{3,i}$ equal to a multiple of the sum of allvalues of that row is a valid row

else it is an invalid row. It is implemented using the formula- $K_{3,j}=(\sum C_{i,j})*k$

Where, 'i' ranges from 1,2,….,n

'k' is a random integer constant that doesn't overflow the 128-bits for $K_{3,j}$

'n' is the number of masked columns in Table T in row 'i'

To find out whether a row is valid or not we check the condition using the formula-

$$R = K_{3,j}\%\left(\sum C_{i,j}\right), \{i = 1,2,3, \ldots\ldots, n\}$$

If R=0 then the row is a true row else it is an invalid row.

Now more false rows we inject onto the table the table's security level increases. But it will require a little extra storage space and also more the rows more number of rows must be scanned and verified by the query which decreases the performance. Therefore we must not inject more false rows into the table.

6. QUERYING THE MASKED E-MOBAT DATABASE

Consider the following table which shows the record of a normal employee database-

Table 1. Original Dataset

ESSN	ENAME	ESAL	EBONUS	EDOJ
111	MARK	20000	8000	12-DEC-12
222	TOM	35000	12000	10-AUG-10
333	ZOYA	30000	10500	23-FEB-11
444	JANE	30000	10000	13-JUL-11
555	GARY	50000	16000	07-DEC-07

Now once the public key $K_{3,i}$ is inserted into the table it transforms into-

Table 2. Original Dataset along with the public key

ESSN	ENAME	ESAL	EBONUS	EDOJ	K_3
111	MARK	20000	8000	12-DEC-12	84333
222	TOM	35000	12000	10-AUG-10	141666
333	ZOYA	30000	10500	23-FEB-11	122499
444	JANE	30000	10000	13-JUL-11	121332
555	GARY	50000	16000	07-DEC-07	199665

The public key $K_3$is equal to three times the sum of all the integer type data in that row. For example, K_3 for 'row 1' is calculated as

$$K3 = 3 * (111 + 20000 + 8000)\{i.e. 3 * (ESSN + ESAL + EBONUS)\}$$

Let the column that has to be masked be ESSN, ESAL and EBONUS.

The value of K_1= 3, $K_{2,ESSN}$ = 300, $K_{2,ESAL}$ = 32000, $K_{2,EBONUS}$ = 12000. Now after applying the masking formula (1), the table transforms itself into-

Table 3. Masked Dataset

ESSN	ENAME	ESAL	EBONUS	EDOJ	K_3
189	MARK	12000	4000	12-DEC-12	84333
88	TOM	3000	1200	10-AUG-10	141666
33	ZOYA	2000	1500	23-FEB-11	122499
134	JANE	1600	2000	13-JUL-11	121332
255	GARY	18000	4000	07-DEC-07	199665

Consider a Query that requests the employeeID(ESSN) and employee name (ENAME)of the employees whose date of joining (EDOJ) is after 10th January 2011 and salary (ESAL) is greater than 9500.

The query that the user writes is-

```
SQL>SELECT ESSN,ENAME
     FROM EMP_DETAILS
     WHERE EDOJ >= '10-JAN-11' AND
ESAL > 9500;
```

Now the E-MOBAT rewrites it as-

```
SQL>SELECT (ESSN-MOD(MOD(K3,i ,3),300) +
300),ENAME FROM EMP_DETAILS
     WHERE EDOJ >= '10-JAN-11' AND ESAL >
(ESAL-MOD(MOD(K3,j,3),32000) + 32000);
```

As shown in the above example the E-MOBAT solely re-writes the queries using the reverse formula(2). These modified queries are not shown to the user. The user only gets the result of the query.

7. SECURITYAND PERFORMANCE IMPROVISATION

7.1 No Access to the Modified Queries for the User

E-MOBAT receives the user queries, modifies it using the keys from the Blackbox, sends the modified query to the DBMS and retrieves the intended results. The modified queries are never visible to the user and for better security purpose the E-MOBAT shuts down all the database historical logs on the DBMS before the execution of the modified instruction because it will disclose the masking keys. The communication between the User Applications, E-MOBAT and the DBMS are encrypted. All these help in increasing the security.

7.2 Non-injective Data Set

Whenever there are two similar values in the database sinceonce we mask the data they turn out to be two different values as the value of K_3 differs for every row. There is another possibility that two different data in the original database can lead to same values after masking since the MOD operator is non-injective. This helps in creating an apparent randomness in the database.

To demonstrate this, consider a table T with two masked columns. Let K_1 = 9264 and $K_{2,1}$ = 12 & $K_{2,2}$ = 78254 for each of the columns. Now when we compare between the original dataset and the masked dataset we can notice the above two conditions. It is demonstrated in Table 4 and Table 5.

Table 4. Different Values in the Original Dataset can give same value in the Masked Dataset

Original Dataset				Masked Dataset		
Col-1	Col-2	$K_{3,i}$		Col-1	Col-2	$K_{3,i}$
11	81873	7550		**22**	1625	7550
20	54129	1898		26	4786	1898
09	71624	5536		15	1198	5536
15	64894	4697		**22**	4657	4697
12	46926	6177		17	4777	6177

Table 5. Same values in the Original Dataset can give different value in the Masked Dataset

Original Dataset				Masked Dataset		
Col-1	Col-2	$K_{3,i}$		Col-1	Col-2	$K_{3,i}$
10	81873	7550		21	4698	7550
21	54129	1898		28	6547	1898
19	71624	5536		**26**	4235	5536
13	64894	4697		16	4488	4697
19	46926	6177		**23**	4135	6177

7.3 Efficiency of the Masking Keys

The Data masking keys play a very important role in our security mechanism. Since K_3 is a public key, only K_1 & K_2 has to be cracked. $K_1$is a 16-byte integer and K_2 has a value varying from 1 to 16-bytes depending on the maximum value of the column. Therefore A minimum of 2^{129} key combinations are required for K_1 & K_2 together thus roughly needs an average number of 2^{128} number of tests for each masked column. If a table has say 'i' number of columns then '$i * 2^{128}$'number of tests are needed to crack the masking key K_2 completely. Now this is a very difficult and time consuming effort. And also we have a mechanism of automatic swapping of K_2 in which the hacker will have to restart his algorithm that was used to crack the key.

7.4 Bandwidth Consumption

Usually in most of the middleware broker data privacy systems the user himself will have to write the queries involving masking and de-masking the data. This results in increased bandwidth consumption. In our solution the E-MOBAT itself rewrites the queries and sends it to the DBMS directly without any involvement of the user. Therefore the bandwidth consumption is reduced.

8. Results& Discussion

In a detailed survey of various databases it has shown that our technique provides a very high security with the balance in the performance. We took a sample Data Warehouse of 100GB in which 80GB was numeric data. We tested the AES128 and 3DES168 algorithms with our technique. We tested it on DBMS Oracle 10g. We compared both the storage and the loading time overheads of both the algorithms with our E-MOBAT using all of the three methods of insertion of the public key, which are-

i) E-MOBAT AddCol: Adding the public key $K_{3,i}$ column in the existing table.

ii) E-MOBAT CreateCol: Recreating the whole fact table along with the public key $K_{3,i.}$

iii) E-MOBAT ExistingCol: Using an existing numeric column to store the public key $K_{3,i}$.

On masking the 80GB numeric data, the following result was obtained

8.1 Storage Space Overheads

In AES128 algorithm the storage space overheads went up to153.9% (123.12GB) and in 3DES168 algorithm it went up to103.6%(82.88GB). When the same was masked using E-MOBAT, while using method (i) the storage space overheads was 5.7% (4.6GB), using method (ii) it was 4.1% (3.28GB) and using method (iii) it was 3.9% (3.12GB). Table 6 depicts the above results.

8.2 Loading Time Overheads

In AES128 algorithm the loading time overheads went up to 189.7% (6337 seconds) and in 3DES168 algorithm it went up to 191.6% (6401 seconds). When the same was masked using E-MOBAT, while using method (i) the loading time overheads was 7.7% (257 seconds), using method (ii) it was 3.5% (116 seconds) and using method (iii) was 3.2% (106 seconds). Table 7 depicts the above results.

8.3 Response Time Overheads

In AES128 algorithm the response time overheads went up to 187%and in 3DES168 algorithm it went up to 203%. When the same was masked using E-MOBAT, while using method (i) the loading time overheads was 35.3% (257 seconds), using method (ii) it was 29.4% and using method (iii) was 22%. Table 8 depicts the above results.

From the above three conditions E-MOBAT gives the best results. Though adding the public key $K_{3,i}$ onto an existing column gives the best result we preferably use the E-MOBAT technique in which the public key $K_{3,i}$ adding is done by recreating the whole table is used.

Table 6. Storage Space Overheads

AES128	3DES168	E-MOBAT (i)	E-MOBAT (ii)	E-MOBAT (iii)
123.12GB (153.9%)	82.88GB (103.6%)	4.6GB (5.7%)	3.28GB (4.1%)	3.12GB (3.9%)

Table 7. Loading Time Overheads

AES128	3DES168	E-MOBAT (i)	E-MOBAT (ii)	E-MOBAT (iii)
6337s (189.7%)	6401s (191.6%)	257s (7.7%)	116s (3.5%)	106s (3.2%)

Table 8. Response Time Overheads

AES128	3DES168	E-MOBAT (i)	E-MOBAT (ii)	E-MOBAT (iii)
+187%	+203%	+35.3%	29.4%	22%

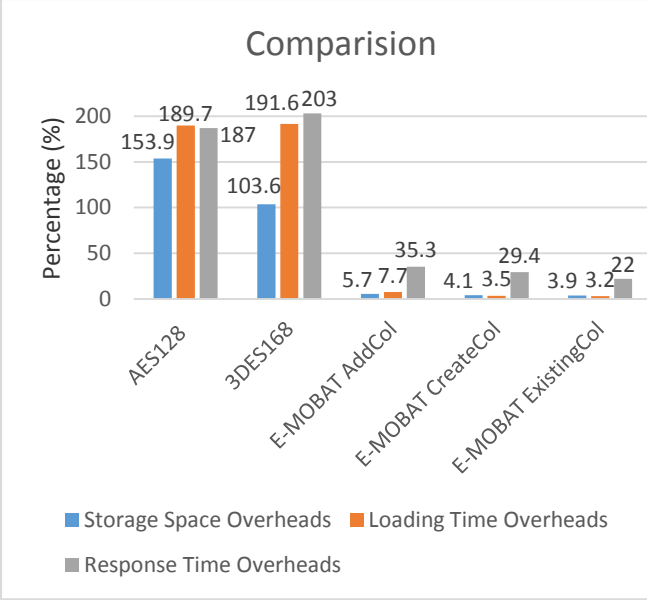

Fig 2: Overheads Comparision

9. CONCLUSION

We propose a Data masking technique that concentrates on enhancement of the security in a Data Warehouse without bringing down its performance or efficiency. To increase the security strength the injection of false rows are also proposed. The Data masking formula used makes use of simple mathematical operators and can be implemented easily on any DBMS. The automatic modification of the query minimizes the efforts of the user and also minimizes the bandwidth consumption. The history log in the E-MOBAT also helps in detecting any intrusions using the pattern stored in it.

We also compared it with the existing technologies like AES128 and 3DES168 algorithms which clearly showed that our security mechanism is much better than them**.**

10. FUTURE WORK

Future work will focus on designing a SQL query segregation. We can design a masking technique for respective data domain.

11. REFERENCES

[1] Ricardo Jorge Santos, Jorge Bernardino, Marco Vieira "Balancing Security and Performance for Enhancing Data Privacy in Data warehouses", CISUC – DEI – FCTUC, 2012.

[2] A Net 2000 Ltd. White Paper "Data masking: What You Need to Know Before You Begin".

[3] Tanya Baccam "SANS Institute Product Review: Oracle Data masking", a SANS Whitepaper, January 2012.

[4] Ravi Kumar G K, Dr. B. Justus Rabi, Dr.Ravindra S. Hegadi, Archana R A "Experimental Study of Various Data masking Techniques with Random Replacement using data volume", International Journal of Computer Science and Information Security, Vol. 9, No. 8, August 2011.

[5] Ravi Kumar G K, Manjunath T N, Ravindra S Hegadi, Umesh I M "A Survey on Recent Trends, Process and Development in Data masking for Testing", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[6] Adam N. R., and Wortmann, J. C. 1989."Security-Control Methods for Statistical Databases:A Comparative study," ACM Computing Surveys (21:4), pp. 515 –556.

[7] Manjunath T.N, Ravindra S Hegadi, Ravikumar G K."A Survey on Multimedia Data Mining and Its Relevance Today" IJCSNS. Vol. 10 No. 11 pp. 165-170.

[8] Manjunath T.N, Ravindra S Hegadi, Ravikumar G K."Analysis of Data Quality Aspects in Datawarehouse Systems", (IJCSIT)

[9] International Journal of Computer Science and Information Technologies, Vol. 2 (1), 2010, 477-485

[10] Dalenius, T., and Reiss, S. P. 1982. "Data Swa`pping: A Technique for Disclosure Control," Journal of Statistical Planning and Inference (6:1),pp. 73-85.

[11] Domingo-Ferrer J., and Mateo-Sanz, J. M. 2002. "Practical Data- Oriented Microaggregation for Statistical Disclosure Control," IEEE Transactions on Knowledge and Data Engineering (14:1), pp. 189- 201.

[12] Epstein, R. A. 2002. "HIPAA on Privacy: Its Unintended and Intended Consequences," Cato Journal (22:1), Spring/Summer, pp. 13-19.

[13] Greengard, S.1996. "Privacy: Entitlement or Illusion?" Personnel Journal (75:5), pp. 74-88.

[14] Kaelber, D., and Jha, A. 2008. "A Research Agenda for Personal Health Records (PHRs),"

[15] Journal of the American Medical Informatics Association (15:6), November / December.

[16] KDnuggets, "Google Subpoena: Child Protection vs. Privacy," Accessed July 2006, from Http://www.kdnuggets.com/polls/2006/google_subpoena .htm.

[17] Liew, C. K., Choi, U. J., and Liew, C. J. 1985."A Data Distortion by Probability Distribution,"ACM Transactions on Database Systems (10:3), pp.395-411.

[18] Xiao-Bai Li, Luvai Motiwalla BY "Protecting Patient Privacy with Data Masking" WISP 2009

[19] Oracle White Paper—Data Masking Best Practices JULY 2010

[20] Sachin Lodha BY Data Privacy – TRDDC Silver Jubilee Commemoration Publication – SL Comments.doc