International Journal of Science and Engineering Applications
Volume 9 - Issue 06, 68 - 78, 2020, ISSN:- 2319 - 7560

An Improved Data Masking Security Solution Using Modulus Based Technique (MOBAT) for Data Warehouse System

Kefas Suwa Larson
M.Sc. Student, Comp. Science
Dept. of Mathematical Science
Abubakar Tafawa Balewa University (ATBU),
Bauchi, Nigeria

Souley Boukari
Professor, Computer Science
Dept. of Mathematical Science
Abubakar Tafawa Balewa University (ATBU),
Bauchi, Nigeria

Abstract: Protecting Data Warehouse (DW) is very important, since it consists of sensitive data used in many business models for decision support processes. Data masking and encryption techniques have been recommended in academic literatures for DW, however, most of the previous works considered only the numeric data types. This research proposes an improved Modulus Based Technique (MOBAT) that supports string data with numeric and non-numeric attributes. Java with NetBeans and MySQL were used in developing the solution. The experimental results show that our suggested technique achieves low storage space overhead, loading time overhead and better processing time performance as compared to the existing system. While existing system consumed 102MB storage space after MOBAT was applied to the data (8% storage space overhead), the proposed methodology maintained the same storage space (1172mb) when data was without encryption and when MOBAT was applied. This means that no extra space is added. Also, loading time overhead for existing system was 7%, but the new scheme achieved 6%. Based on the experimental results obtained, our technique is more efficient, making it a valid alternative for protecting data stored in DWs.

Keywords: Data Warehouse Security; Data Warehousing; Data Security; Data Masking; Data Encryption

1. INTRODUCTION

Although many studies have been conducted on security issues in Data Warehouse (DW), attacks are increasing every year in numbers and complexity, and none of the security solution proposed so far by researchers has proffered a permanent and all-inclusive answer to the data damage, breaches, malicious attacks, etc.

Security concerns have been an important subject in the corridor of DW system and it remains totally unresolved up to date **[1]**. The data stored in warehouse is extracted from various operational databases. So, security has always been the critical issue in DW for protection of essential and useful data **[2].** Due to the widespread use of confidential data in Data Warehousing systems, security is a major concern **[3]**.

According to **[4],** the main concept of DW Security is about Confidentiality, Integrity and Availability (CIA). Confidentially means only authorized users should access the data from the DW. Integrity means originality of the data. Availability means information is available all the time**.** Authors in **[5],** indicated that to comply with the CIA attributes, many techniques have been submitted. These can be classified into two broad categories: preventive and reactive techniques. Preventive data security techniques protect the data in advance from security breaches or attacks. They used data masking, encryption, and data access policies to tackle the preventive category. Reactive data security techniques effectively respond after a security attack or security problem has occurred**.**

Granting that a variety of standard encryption algorithms are available to secure DW, but as a consequence, they do reduce the performance of the DW system due to their required large computational overheads **[6].** Most of the DW Security approaches used encryption and masking methods that tried to provide strong data privacy. However, these types of encryption method make them inefficient for DW use owing to their high computational overheads. Therefore, a data masking technique is needed that can provide strong data privacy with less computational effort and also maintains high performance **[7].**

Furthermore, researchers in the past have used masking and encryption techniques to protect data in DW, however, most of the previous researches applied the masking and encryption techniques to sensitive numeric attributes only. In this study, an improved Modulus Based Technique (MOBAT) that supports string data with textual, alphanumeric, numeric and special character attributes specifically designed for DW system is proposed.

The rest of this paper is organized as follows: Section **2** describes review of related works. Section **3** presents the new MOBAT methodology. Section **4** presents the results of the implementation of new MOBAT and the simulation conducted, and finally, Section **5** concludes the study and highlights future research directions.

2. REVIEW OF RELATED WORKS

This section provides a review of related works done by different researchers on DW Security, pointing out the strengths and weaknesses of each proffered solution.

The authors in **[8],** proposed a solution based on user profile, which considers the definition of access permissions according to the user role using the access rights defined in data sources, generate the level of sensitivity of each object in the DW according to these permissions, then trace the access and detect violation attempts of access rights on a sensitive data. They claim that this technique helps the owner of the DW to well manage the access control of the users. But this type of solution can only be suitable for applications with a

limited number of users and roles and where the user's roles seldom change, not for volume-centric data environment such as DW.

[1] presented a framework for securing a student DW by creating a hybrid technique using email and password, Token and CAPTCHA authentication to ensure that only registered students have access to the DW. However, their solution only restricts logging access to the DW, while data at rest is not masked, hence making it extremely vulnerable to a breach.

In their work, **[9]** evaluated a new schema that balances security and performance when outsourcing DW in the cloud. The schema is based on a simple privacy homomorphism using the MOD operator with 2 prime numbers p and q. The scenario is to encrypt data stocked in DW with the encryption function $\phi(x)= [x \bmod p, x \bmod q]$ *and* $m = pq$ (the product of these large secret primes). Unfortunately, the schema is not secure enough because the cloud provider can infer the two chunks of data and get the two secret parameters p and q. Malicious intruders can also break the security parameters of the cloud provider, get the encrypted data and the modulo m from the cloud provider and decrypt the data using the known clear text.

[2] discussed an enhanced security architecture for DW by combining One-Time Pad (OTP) encryption technique with Advanced Encryption Standard (AES) to encrypt the data before loading it into the DW. The OTP encrypts the input data with random key (k) using modular addition, mod26 which has tremendous properties that plays an essential part in cryptography for security. The OTP is unbreakable theoretically but practically it is weak.

In their paper**, [10**] considered a solution that uses a universal scheme for hiding data of various type fields of a row (tuple) of a database table based on the use of MOBAT public and private keys (K1, K2 and K3). Their method is based on random permutation of elements (bytes, characters) of data of a specific field of a different type (numeric, character strings, Binary Large Objects (BLOBs), Character Large Objects (CLOBs)) of table row, which used data shuffling technique. However, only 18,000 rows of data were subjected to the masking and encryption evaluation, thus, may not be sufficient for a DW that stores huge volume of data.

In **[11**], the authors discussed a framework to Identify, Map, Apply, Sign, Keep testing, and Utilize (IMASKU) and Content-Based Data Masking Technique to securely save sensitive data into an integrated DW to prevent the database from the risks of external and internal attacks. Their technique is claimed to protect data at rest within the enterprise data warehouse. However, further algorithm optimization method is needed to determine the acceptable execution time when a big size of data is to be used on the framework.

[12] presented a new and efficient Format Preserving Encryption (FPE) scheme for encrypting integer data of 16 digits by using AES, exclusive OR operation and a translation method to overcome the shortcomings of existing schemes. However, the technique only covers 16-digit numeric data and may not be feasible for DW system that contains various data types.

[13] considered a Multivalued-Homomorphic (MV-HO) encryption strategy that was compared with encryption strategies based on symmetric encryption, order preserving symmetric encryption and homomorphic encryption. They claim that their technique is the best solution as it is pareto-optimal with respect to other strategies investigated. But the technique was only tested with a small size of data, which is not characteristic of a DW that holds huge volume of data.

To deal with numeric and non-numeric data types, **[14**] proposed a technique to protect the confidentiality of numeric and non-numeric data by obfuscation and encryption before storing into the Cloud storage. They claim that their technique has reduced the service cost, minimized the data size and processing time while uploading into the cloud storage. However, there was no experimental evaluation carried out to ascertain this assumption.

[15] evaluated a new framework for implementing security issues in DWs named DW Signature (DWS). The DWS framework focuses on the triage of security issues, which are Confidentiality, Integrity and Availability (CIA). Their approach achieves high performance by using parallel computing through a middleware named View Manager Layer (VML). However, the method lacks performance evaluation of (i) finding the query memory buffer for the VML middleware, and (ii) evaluating the high performance when the number of executors increases in the VML middleware.

In **[16],** the authors advanced an integrated data security framework that enables the use of data masking, encryption and intrusion detection in a single workflow for DW environment. The framework discusses the feasibility issues involving solutions that promote data confidentiality and deal with intrusions against DWs at the database level, focusing on data masking, encryption and database intrusion detection systems (DIDS). However, both the data masking and encryption techniques presented were specifically designed to mask and encrypt numeric values only. This is because of the believe that in most DWs the main portion of sensitive data is numerical. Thus, the solution does not cover all the sensitive data that may be stored in DW.

[17] presented "An Effective DW Security Framework" which highlights the usage of modulus operator in data security for DW system. They replaced 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 the modulus operator. They also injected false rows onto the database which uses up extra space but helps in increasing the randomness in the database that can mislead the hackers. Their technique was compared with standard encryption algorithms such as AES128 and 3DES168. The results gotten were highly favorable with balanced system performance in storage space overheads, loading time overheads and response time overheads. However, there is need to design a masking technique for respective data domain (since DW is made of data from different sources) so that it can apply to all data types instead of the numeric data only.

[18] proposed a Specific Encryption Solution for DW (SES-DW) using only standard SQL operators such as eXclusive OR (XOR) and modulo (MOD, which returns the remainder of a division expression), together with additions and subtractions. Their technique shows a better database performance than standard and state-of-the-art encryption solutions with security based on DW perspective. However, its major shortfall is that it can only secure sensitive data with numeric data types. It does not cover other data types such as textual or alphanumeric characters.

The authors in **[19**], presented a lightweight encryption technique based on a cipher using alternating sets of eXclusive OR (XOR) and bit switching operations sequence

which focuses on leveraging security-performance tradeoffs that can make it feasible for DW environments. But, the viability and feasibility of the technique could not be guaranteed since the method was not tested in a real-world DW environment.

[20] implemented a model for securing data in DW based on log implementation. They noted that, data stored in DW need to be transformed to other form which should be unreadable to attackers. Thus, to increase data security the authors have suggested a technique called data masking, which is a process to convert original data to some other form. For this, a MOD function/operator is used in SQL such that whenever a user sends request stored in a log, it is verified in the log and resent. But their data masking method introduced large overheads, making it unfeasible for DW environment.

[21] introduced a Big Data Security mechanism using the MOBAT technique, wherein they first selected only certain attributes that have higher values than the rest and secure them, which in turn provided security to the whole of the Big Data. To mask the numeric data in the Big Data they made use of the mathematical formula with MOD (modulus) operator (which returns the remainder) and a set of other basic arithmetic operators. However, the technique covers only numeric data. There is need to extend the focus to securing the character data as well.

[22] reviewed the security measure to prevent sensitive data from malicious attacks. He provided a log-based security system architecture to prevent the data from attackers. But the critical problem is on how to automatically coordinate the access rights of the DW with those of the data from the different sources.

[23] presented a paper on how to balance Security and Performance for Enhancing Data Privacy in DWs using MOdulus-BAsed method. Their proposal uses the MOD operator and simple arithmetic operations to mask data and provide a significant level of apparent randomness for the masked values. The solution also uses one of the masking keys for injecting false data into the DW in order to mislead attackers and increase the overall security level, making them unable to distinguish true from false data. But their masking technique is also only tailored to numeric kind of data.

In **24]**, the authors presented the best database encryption solutions to protect sensitive data. They proposed a data masking solution for numerical values in DWs based on the mathematical modulus operator, which can be used with an extra software application layer. However, this technique needs to be expanded to cover masking of alphanumeric values, so that it can provide a complete data protection solution.

3. PROPOSED METHODOLOGY

This section presents the expected methodology used in securing the data in a DW system.

In our proposed solution, we have designed a model to implement one of the future works of **[16]**, in which the data protection was only for numeric data type. The extension of the existing techniques to cover all the data types is imperative because DW stores huge amount of sensitive data types that must be guarded against from both inside and outside attackers.

The main contribution of our research work is the expansion of the existing MOBAT technique to ensure all sensitive data with different data types in DW can be encrypted, and to prevent unauthorized access. This was designed and extended using the 95-Printable ASCII codes for alphanumeric characters and symbols, while maintaining the existing MOBAT formula for the numeric data type. Figure 1 depicts the framework of the improved security solution.

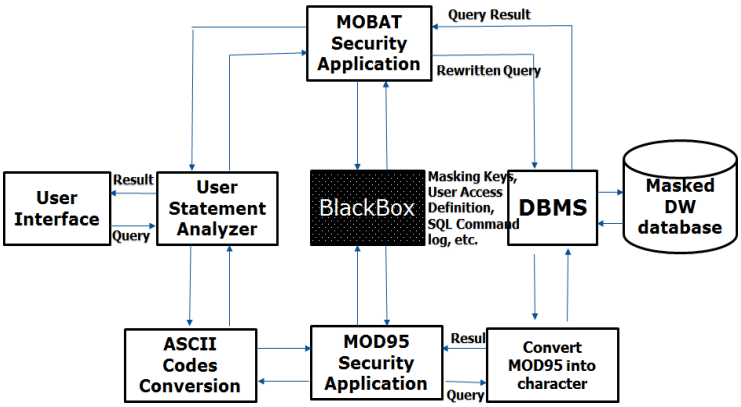

Figure 1. Proposed MOBAT and MOD95 system with extended components

3.1 Working Principle of the Proposed System

This research proposes an improved MOBAT for securing data in Data Warehouses based on **[16]**. Our technique is a hybrid of MOBAT and MOD 95 which uses the 95 printable ASCII codes table that contains all the input on a normal computer keyboard. This provides for easy conversion of almost all English characters you can find on most computer systems, and enables encrypting all data types. The considered structure is mainly divided into two modules, numeric and non-numeric data masking.

3.1.1 Numeric data masking

To better understand the working of the numeric data masking technique, let us consider a table 'T' with a set of 'm' rows and 'n' columns, where rows are given by (R1, R2,……,Rm) and columns are given by (C1, C2,……,Cn). Let the value that has to be masked be represented as a pair of (Ri, Cj) where 'Ri' is the row and 'Cj' is the column which contains the data that has to be masked. In order to mask the data, we must have three masking keys:

- K1, a 128-bit randomly generated number which remains constant for the table T.
- K2, a 128-bit randomly generated number which remains constant for a particular column Cj. This is represented as K2, j.
- K3, a random 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 K3, i.

K1 and K2 are stored in the Black Box, while K3 will be stored in the masked table along with the other data. Thus, every row of a table will compulsorily have a public key K3.

Now, let's suppose a value to be masked as (Ri, Cj). Then the new masked value (Ri, Cj)` is given by the formula:

$$(Ri, Cj)' = (Ri, Cj) - ((K3, i \bmod K1) \bmod K2, j) + K2, j \quad (1)$$

To retrieve the original value, we used the following formula:

(Ri, Cj) = (Ri, Cj)' + ((K3, i Mod K1) Mod K2, j) - K2, j (2)

3.1.2 Non-numeric data masking

For the non-numeric data types, we use a simple linear algebraic function with the private key (K2) to encrypt them by employing the formula stated in **[25]**:

E(x) = ((x - 32 + K2 + 95) mod 95) + 32 (3)

where x is the equivalent value of each character to mask/encrypt as contain in ASCII table, while the randomly generated key K2, is column dependent. We are subtracting 32 from every element of x because the printable characters have ASCII codes in the range of 32 to 126, while mod 95 represents the ASCII values in the range 0 to 94.

To reverse any of the encrypted characters, we have to reverse the steps used in encrypting it to recover the original values.

D(E(x)) = ((x - 32 - K2 + 95) mod 95) + 32 (4)

Thus, to reverse any of the encrypted characters, we have to reverse the steps used in encrypting it to recover the original values.

3.1.3 Proposed program design

The main algorithm of the submitted solution is as shown in Figure 2, while details of the split algorithms are numbered 1 to 6 subsequently.

Begin algorithm
1. For each table n in the target DW database TDW(1….N)
2. Fetch K1 private key common for the entire table(n)
3. For each attribute j in the table
4. Fetch K2,j private key common for entire attribute(j)
5. For each sensitive data item (Ri, Cj) selected from the table(n)
6. Fetch K3,i public key common for a tuple (i)
7. **If Data item selected is NOT numeric**
8. **Convert each character to its ASCII equivalent and store in E(x)**
9. **Encrypt/ Decrypt E(x) with MOD95 masking formula**
10. **Translate E(x) to its character equivalent (ciphertext)**
11. **Append E(x) to (Ri, Cj)**
12. **end for**
13. else
14. for each pair of digits selected in (Ri, Cj)
15. encrypt/ decrypt digit value with MOBAT masking formula and store in numb
16. append numb to (Ri, Cj) //every loop appends with previous values
17. end for
18. end for
19. end for
20. end algorithm.

Figure 2. Algorithm for DW Security using MOBAT and MOD95

1. Algorithm for Encryption with MOBAT and MOD95
Step 1: Identify sensitive Data to mask
Step 2: If data is Numeric Then
Step 3: Encrypt data using MOBAT algorithm
Step 4. Go to Step 9
Step 5: If Data is Non-numeric THEN
Step 6: Convert character to ASCII code
Step 7: Encrypt data using MOD95
Step 8: Convert the MOD95 into character to produce ciphertext
Step 9: Store Data in DW database

2. Algorithm for Decryption with MOBAT and MOD95
Step 1: Identify encrypted Data to unmask
Step 2: If data is Numeric Then
Step 3: Decrypt data using reverse formula of MOBAT algorithm
Step 4: Go to Step 9
Step 5: If Data is Non-numeric THEN
Step 6: Convert ciphertext to ascii code
Step 7: Decrypt data using the reverse formula of MOD95
Step 8: Convert the MOD95 into character to produce plaintext
Step 9: Store Data in DW database
Step 10: Display result (original data) to user

3. MOBAT Encryption Algorithm for Numeric Data types
Step 1. Select Numeric data fields to encrypt, in our case (l_quantity, l_extendedprice, l_discount and l_tax)
Step 2: Fetch masking keys (k1, k2)
Step 3: Generate k3 for each row
Step 4: Apply MOBAT formula to data
Step 5: Store record in DW masked database
Step 6: Repeat steps 2 to 5 for each row until end
Step 7: Calculate execution time (in second).
Step 8: Display result to user

4. MOBAT Decryption Algorithm for Numeric Data types
Step 1. Select Numeric data fields to decrypt, in our case (l_quantity, l_extendedprice, l_discount and l_tax)
Step 2: Fetch masking keys (k1, k2)
Step 3: Generate k3 for each row
Step 4: Apply reverse MOBAT formula to data
Step 5: Repeat steps 2 to 4 for each row until end
Step 6: Calculate execution time (second)
Step 7: Display results to user

5. MOD95 Encryption Algorithm for Non-Numeric Data types
Step 1: Select non-numeric data to encrypt, in our case (l_shipmode)
Step 2: Fetch k2 for each column
Step 3: Convert data to ascii code equivalent
Step 4: Apply MOD95 formula to data
Step 5: Convert MOD95 into character to produce Ciphertext
Step 6: Store record in DW masked database
Step 7: Repeat steps 2 to 6 for each row until end
Step 8: Calculate execution time (in second)
Step 9: Display results to user

6. MOD95 Decryption Algorithm for Non-Numeric Data types
Step 1: Select non-numeric data field to decrypt, in our case (l_shipmode)
Step 2: Fetch k2 for each column
Step 3: Convert data to ascii code equivalent
Step 4: Apply MOD95 reverse formula to retrieve original data
Step 5: Convert MOD95 into character to produce Plaintext
Step 6: Repeat steps 2 to 5 for each row until end
Step 7: Calculate execution time (in second)
Step 8: Display results to user.

3.1.4 Functional flowchart of the proposed work

The working of the improved MOBAT is further depicted by the Flowchart in Figure 3.

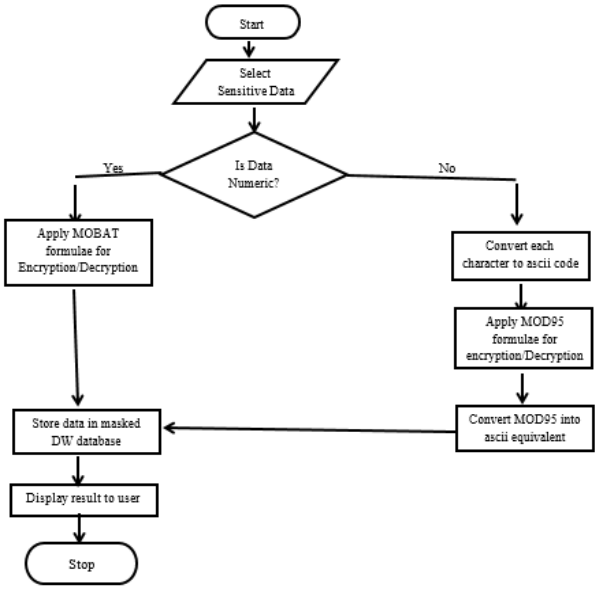

Figure 3. Flowchart of the Proposed System

3.1.5 Evaluation metrics

The evaluation of the masking /unmasking process is conducted using a dataset downloaded from the TPC-H database. Two data masking metrics are considered:

Storage Growth – The change in the size of new masked LineItem data table in comparison with the original one measured in megabytes (MB).

This is measured as follows:

$$\text{Storage Size Overhead (\%)} = \frac{\text{Storage size in MOBAT (MB)} - \text{Storage size in MySQL (MB)}}{\text{Storage size in MySQL (MB)}} \times 100$$

Performance – Loading time overhead (%) and the processing speed of the encryption/decryption algorithms in seconds. A time measurement is performed to evaluate the execution cost of data loading operation in standard application compared to loading time in the proposed approach (MOBAT).

This is measured as follows:

$$\text{Loading Time Overhead (\%)} = \frac{\text{Loading execution time with MOBAT (Sec)} - \text{Loading Time in MySQL database (sec)}}{\text{Loading Time in standard database (sec).}} \times 100$$

3.1.6 Masking key management

A DW encryption solution is only as secure as the protection of its encryption keys. Therefore, the way in which encryption keys are accessed, restricted and stored is critically important. As stated in **[16]**, the public key K3, is stored in the fact table, so only keys K1 and K2 need to be cracked in MOBAT. K1 is a 16-byte integer, that is, a set of 128 bits. K2 depends on maximum storage size defined for each column, but variable between 1 and 128 bits. This means our technique is a minimum of 2^{129} key combinations for K1 and K2 together (at least 16 bytes+1 bit), and roughly needs an average number of 2^{128} tests (half of the total possible brute force tests – 50% chance) for discovering the keys using brute force, for each masked column in the table, since K2 is column dependent. For example, the minimum number of combinations needed to discover all key values for an i^{th} number of columns is $i * 2^{129}$, resulting in an average of $i * 2^{128} \approx i * 3.4 \times 10^{38}$ brute force tests to discover the keys. As stated in **[11]**, the maximum time needed to crack these masking keys versus the 128-bit key combination is 1.02×10^{18} years (1 billion years). This is a very difficult and time-consuming effort given the high number of possible brute key values to crack. Also, the MOBAT algorithms use dual moduli for the encryption and decryption process, thereby providing strong security against Brute-force attacks **[26].**

4. RESULTS AND DISCUSSION

In this section, we analyzed and discussed the contemplated data masking solution which is specifically designed to improve data confidentiality in DWs.

4.1 Simulation Environment

The work is implemented using Java programming language with Java development kit (JDK) 1.8 and NetBeans IDE 8.2 connected to XAMPP-MySQL DBMS on an intel Pentium N3540 Processor, 2.16GHz CPU with a 500GB hard disk and 4GB RAM. Various data sample testing was conducted to determine the execution time of the algorithms, and to know the storage space consumed when different size of data is loaded into the DW database**.**

4.2 Experimental Data

To assess performance of the new masking technique, we used dataset of a DW Fact table called LineItem that was extracted from the Transaction Processing Council ad-hoc (TPC-H) decision support benchmark to test the encryption/decryption algorithms. A brief description of the LineItem Fact table can be found in Appendix A.

4.3 Dataset

The dataset used in this research work was downloaded from the TPC-H decision support benchmark [TPC-H]. The data schema of TPC-H is created as sales DW with one fact table (LineItem), joint by seven-dimension tables on a standalone laptop.

From the sample dataset, we selected four sensitive numeric attributes and one non-numeric attribute and applied MOBAT and MOD95 algorithms respectively (L-Quantity, L_ExtendedPrice, L_Discount and L_Tax) and L_Shipmode). Also, for the experimental encryption, we tested one scenario, namely: the results for MOBAT where the public key K3, i columns are added to the fact table (LineItem) before encrypting any of the sensitive column, named as MOBAT AddCol. Finally, the results of the application of MOBAT and MOD95 masking formulae (1 & 3) on normal data and encrypted data are compared based on the scenario stated in Table 1.

Table 1. Experimental encryption /masking scenario

Reference/ Label	Description
MOBAT AddCol	Data masked with MOBAT formulae (1 & 3) in which a column for masking key k3,i has been added to the fact table

4.4 Presentation and Analysis of Results

This section discusses the operation performed and the time it takes to encrypt and decrypt data using the proposed MOBAT and MOD95 algorithms. After conducting four (4) test cases of simulation, the summary of results obtained are shown in Table 2. The complete set of test results and respective statistical measures can be seen in Appendix B.

Table 2. Comparison of experimental results

Test Case	No. of Records	Data Size (MB)	Operation	MOBAT Algorithm (Sec)	MOD95 Algorithm (Sec)
Case 1	500,000	78	Load Time (sec)	44	
			Encryption Time (sec)	43	37
			Decryption (sec)	34	30
Case 2	1,000,000	144	Load Time (sec)	90	
			Encryption Time (sec)	87	76
			Decryption (sec)	80	79
Case 3	1,500,000	223	Load Time (sec)	139	
			Encryption Time (sec)	152	134
			Decryption (sec)	147	133
Case 4	3,000,000	440	Load Time (sec)	273	
			Encryption Time (sec)	321	295
			Decryption (sec)	282	261

4.4.1 Analysis of experimental results

4.4.1.1 Test case 1 using 78mb data

Figure 4 shows the graphical representation of the storage space used, loading time of the extracted data and the processing time while encrypting 78MB of data.

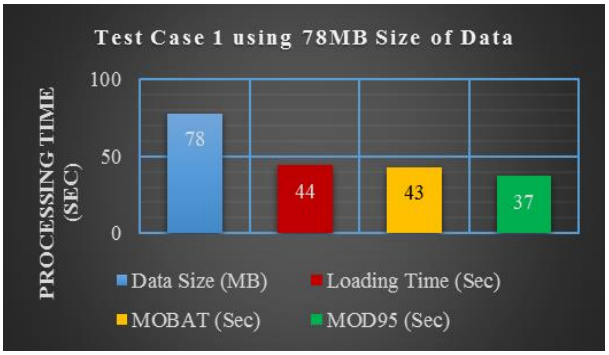

Figure 4. Processing time executed in Test case 1

4.4.1.2 Test case 2 using 144mb data

Figure 5 shows the graphical representation of the storage space used, loading time of the extracted data and the processing time while encrypting 144MB of data.

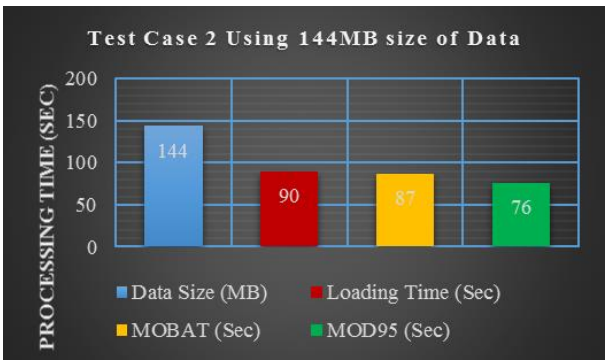

Figure 5. Processing time executed in Test case 2

.

4.4.1.3 Test case 3 using 223mb data

Figure 6 shows the graphical representation of the storage space used, loading time of the extracted data and the processing time while encrypting 223MB of data.

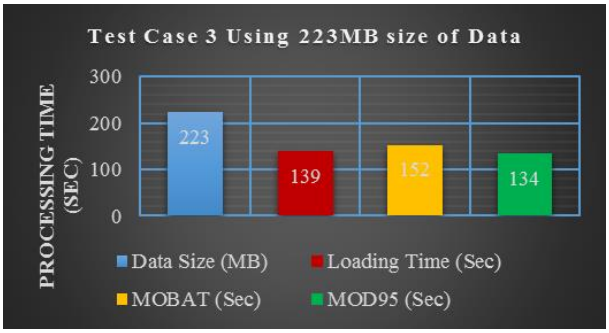

Figure 6. Processing time executed in Test case 3

4.4.1.4 Test case 4 using 440mb data

Figure 7 shows the graphical representation of the storage space used, loading time of the extracted data and the processing time while encrypting 440MB of data.

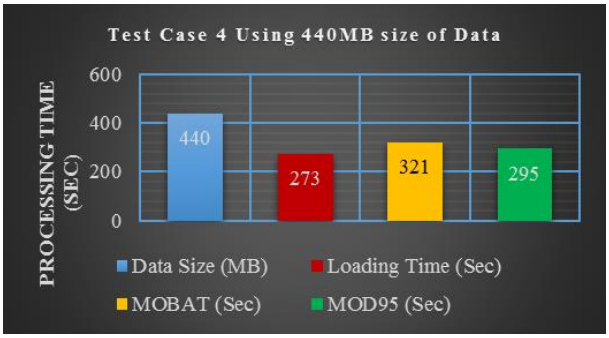

Figure 7. Processing time executed in Test case 4

4.4.2 Analysis of storage space

This section describes the total data storage space used (in MB) and the percentage storage overhead after loading the TPC-H 1GB LineItem Fact table into the DW. We analyzed both the standard storage space used by the LineItem fact table when it is without any sort of encryption and when the columns of the fact table have been masked, as can be seen in Table 3. This is to find out whether the application of MOBAT and MOD95 processes will write additional data that may take up extra storage space in the DW database.

Table 3. Storage size and overhead for the TPC-H 1gb tables.

	Proposed MOBAT		Existing MOBAT	
Test Case	Standard MySQL	MOBAT (MB)	Standard SQL Server	Existing MOBAT
Test Case 1	78	78	1237	1339
Test Case 2	144	144	1237	1339
Test Case 3	223	223	1237	1339
Test Case 3	440	440	1237	1339
Dimensions	287	287	1237	1339
Total Storage Size (MB)	**1172**	**1172**	**1237**	**1339**
Storage overhead (%)		0%		8%

Additionally, from Table 3, the existing system takes up 1237mb of storage space when it is without any encryption. But when MOBAT was applied to the data it consumed 1339mb of storage space**.** This means there is an increase of 102MB storage space after the MOBAT was applied to the data, leading to 8% storage space overhead.

However, in our considered MOBAT the storage space consumed when data was loaded without any encryption and when MOBAT/MOD95 was applied is the same, 1172mb. This means that no extra space is added when the proposed MOBAT was applied to the data.

Another key observation worth noting is the significant difference between the total data storage space sizes of SQL Server (1237MB) and MySQL (1172MB) as can be seen in Figure 8. The huge difference in the standard data storage space sizes between these DBMS is because they have distinct ways of storing data [16]. Research has shown that MySQL with MariaDB can improve compression performance for flash devices, improves storage efficiency, and improve power efficiency and CPU utilization **[27].**

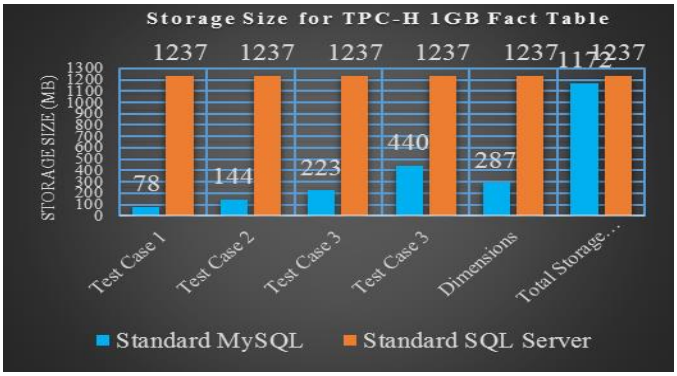

Figure 8. LineItem fact table storage size (MB)

Figure 9, shows that the improved MOBAT has incurred **0%** overhead in storage space when compared to the 8% recorded in the existing MOBAT. This means a huge cost savings in memory usage when using the novel solution. Figure 9, shows that the enhanced MOBAT has incurred **0%** overhead in storage space when compared to the 8% recorded in the existing MOBAT. This means a huge cost savings in memory usage when using the new MOBAT. The storage overhead is evaded by preserving each of the encrypted column's data type **[18].** This is achieved by retaining the original column-type and length of each encrypted column. This ensures the encrypted data is realistic but not real, and enables generating accurate but not factual results.

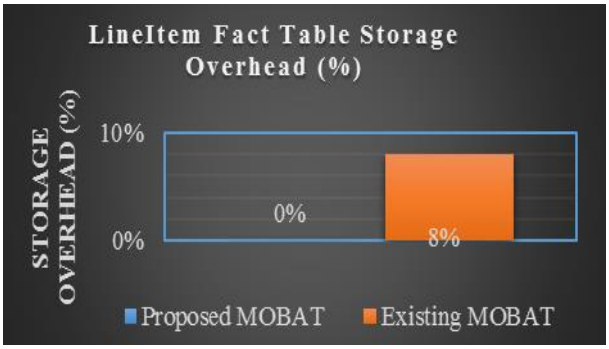

Figure 9. LineItem storage space overhead (%).

4.4.3 Analysis of load time

In this section, we analyze the loading time taken when populating the fact table to know how long the execution of the data size by the MOBAT solution takes. Table 4 shows the Loading time of both the existing system and the improved MOBAT.

Table 4. Lineitem fact table loading time (sec) and loading time overhead (%).

	Existing MOBAT		Proposed MOBAT	
Test Case	**Standard SQL Server (Sec)**	**MOBAT Addcol (Sec)**	**Standard Mysql (Sec)**	**Proposed MOBAT (Sec)**
Test Case 1	212	227	34	44
Test Case 2	212	227	84	90
Test Case 3	212	227	134	139
Test Case 4	212	227	262	273
Total Loading Time	212	227	514	546
Loading Time Overhead (%)		7%		6%

Figures 10 and 11 respectively show the results of the total loading time (in seconds) and the percentage of time overhead (%) for updating the TPC-H 1GB *LineItem* fact table. It can be observed that the total standard loading time for the LineItem fact table without using any sort of encryption solution is 212 seconds, and after MOBAT has been applied the loading takes 227 seconds as can be found in **[16]**. For the proposed system, total standard loading time takes 514 seconds, while the loading time when new MOBAT/MOD95 is applied takes 546 seconds.

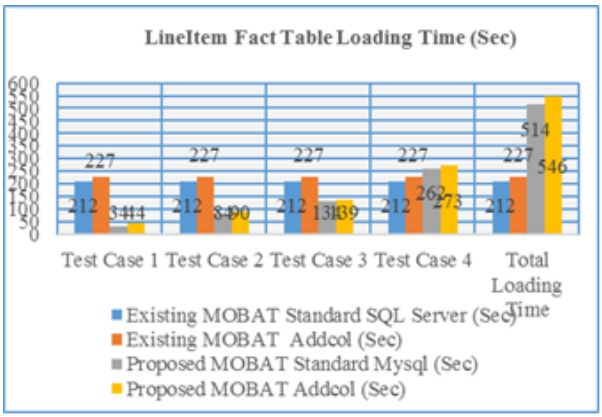

Figure 10. LineItem fact table loading time (sec).

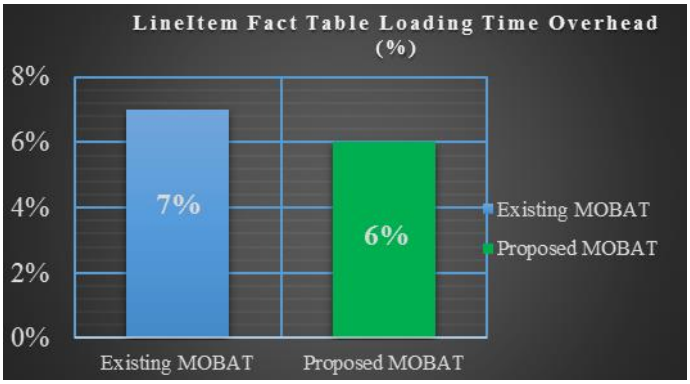

Figure 11. LineItem fact table loading time overhead (%).

A key observation that we can make from Figure 11 is the loading time overheads. While the existing system has loading time overheads as 7%, in the improved MOBAT it is 6%. This clearly shows that our MOBAT performs better than **[16].**

4.4.4 Analysis of processing time

This section looks at the encryption and decryption speed for the four test cases to determine how long it takes MOBAT and MOD95 algorithms each to complete its masking task. Thus, the processing time of the encryption and decryption processes while updating the sensitive data is analyzed as shown in Figures 12 and 13.

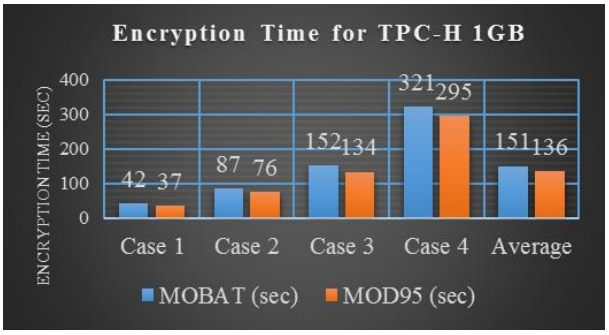

Figure 12. Encryption time (sec) for the TPC-H 1gb fact table per algorithm.

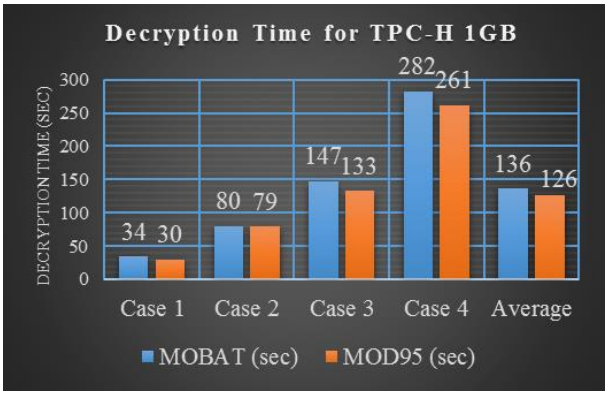

Figure 13. Decryption time (sec) for the TPC-H 1GB fact table per algorithm.

4.5 Discussion on the Experimental Results

4.5.1 Storage space overheads

The 1172MB storage space consumed in the new system when data was loaded without any encryption and when MOBAT and MOD95 algorithms were applied shows that there is no extra space added to the system. This none increase in the storage size for the proposed system is explained by the fact that the improved MOBAT algorithm preserves the encrypted columns' data type format **[28]**. Thus, it avoids introducing storage space overhead and type conversions while running the encryption process.

Similarly, while the novel solution involves **0%** as storage space overhead, the existing system had 8% storage space overhead. This is because in the new entity the total storage space size did not change in both MySQL and when the data was masked with the new MOBAT. This signifies an efficient performance of the developed MOBAT.

4.5.2 Loading time overheads

The loading time overheads of 6% achieved by suggested system is a significant improvement compared to the 7% recorded in **[16].**

4.5.3 Processing time performance

4.5.3.1 Encryption time when masking data

As can be seen in Table 3 and Figure 12, each of the processing speed of MOBAT and MOD95 is directly proportional to the data size. That is, as the size of data increases so does the execution time of the algorithms. In the first, second, third and fourth test cases conducted, MOBAT algorithm took 42, 87, 152 and 321 seconds to encrypt data size of 78MB, 144MB, 223MB and 440MB, while MOD95 progressively took 37, 76, 134 and 295 seconds to encrypt the same size of data.

4.5.3.2 Decryption time when unmasking data

For the decryption process (Figure 13), the first, second, third and fourth test samples for MOBAT took 34, 80, 147 and 282 seconds, while MOD95 took 30, 79, 133 and 261 seconds to decrypt the same size of data.

A further observation from Figures 12 and 13, is that the processing time of MOD95 is lower than MOBAT. The MOBAT algorithm takes longer time to encrypt/decrypt data than MOD95 because in the experiment we chose four columns (l_quantity, l_extendedprice, (l_discount, and l_tax) to encrypt/decrypt with MOBAT while only one column (l_shipmode) was used to test the MOD95 algorithm.

4.6 General Observation

An improved data masking solution specifically designed for ensuring data confidentiality in DW was developed. The advanced data masking formulae are composed by a set of two consecutive moduli (division remainder) operations and some simple arithmetic operations.

Considering the results attained from the experimental tests, it is clear that the enhanced MOBAT is much more efficient. The recommended technique introduces low storage space overheads, low loading time overheads and better processing time performance in the system. Precisely, the proposed MOBAT is much faster than the existing solution, introducing 6% vs 7% of loading time and zero percent storage space overheads in the tested scenarios. The experimental results have demonstrated that the solution can effectively be used as a valid option for protecting sensitive data in Data Warehouses.

5. CONCLUSION AND FUTRE WORK

An improvement over existing MOBAT has been achieved, by allowing all string data types (textual, alphanumeric, special characters and numbers) to be masked, thus guaranteeing data privacy and confidentiality of data stored in data warehouses.

In this improved version of MOBAT, ASCII codes are used to encode all the 95 printable characters, thereby adding advantage of easy processing and implementation. That is, we have increased the scope of both data masking and encryption techniques to cover the protection of textual and alphanumeric attributes, apart from numerical attributes. The keys used in the encryption are randomly generated and so help to encrypt

the data that is required continuously by producing different values for different columns and rows of data.

The experimental results show that the improved MOBAT and MOD95 algorithms successfully encrypted all string data types in the sample data. This performance also comes without compromising the database size. The storage space overheads, loading time overheads and processing time performance introduced by our refined technique is lower than that in **[16].** This allows us to state that our improved MOBAT solution is a viable data security alternative that can be used to secure all string data in DW.

In the future, this approach can be expanded to cover all data types that may be stored in a DW such as metadata (XML), image, pdf, audio/video, etc. We also plan to

• implement and analyze the query performance of the encryption algorithms using some of the 22 TPC-H benchmark queries.

• simulate the practicability, efficiency and effectiveness of the new solution in a real-world Data Warehousing environment.

6. REFERENCES

[1] Kalio, Q. P., & Nwiabu, N. D. (2019). A framework for securing data warehouse using hybrid approach. International Journal of Computer Science and Mathematical Theory, 5(1), 44-55.

[2] Gupta, S., Jain, S., & Agarwal, M. (2019). DWSA: A secure data warehouse architecture for encrypting data using AES and OTP encryption technique. in soft computing: Theories and Applications (Vol. 742, pp. 505-514): Springer.

[3] Homayouni, H., Ghosh, S., & Ray, I. (2019). Data warehouse testing. in advances in Computers (Vol. 112, pp. 223-273): Elsevier.

[4] Kumar, S., Singh, B., & Kaur, G. (2016). Data warehouse security issue. International Journal of Advanced Research in Computer Science, 7(6).

[5] Divya, K., & Kurmi, J. (2017). A reassessment on security tactics of Data Warehouse and comparison of compression algorithms. Advances in Computational Sciences and Technology, 10(5), 847-854.

[6] Chandra, P., & Gupta, M. K. (2018). Comprehensive survey on data warehousing research. International Journal of Information Technology, 10(2).

[7] Phoghat, P., & Maitrey, S. (2015). Analysis of security techniques and issues in Data Warehouse. Paper presented at the 2015 1st International Conference on Next Generation Computing Technologies (NGCT).

[8] Elouazzani, A., Harbi, N., & Badir, H. (2018). User profile management to protect sensitive Data in Warehouses. *9*(1), 1-32.

[9] Karkouda, K., Nabli, A., & Gargouri, F. (2019). TrustedDW: A new framework to securely hosting data warehouse in the Cloud. Proceedings of 34th International Confer, 58, 397-406.

[10] Yesin, V. I., & Vilihura, V. V. (2019). Some approach to data masking as means to counteract the inference threat. Radio Engineering, 3(198), 113 -130.

[11] Ali, O. (2018). Secured data masking framework and technique for preserving privacy in a business intelligence analytics platform. Electronic Thesis and Dissertation Repository. 5995.

[12] Gupta, S., Jain, S., & Agarwal, M. (2018). Ensuring data security in databases using format preserving encryption. Paper presented at the 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence).

[13] Lopes, C. C., Cesário-Times, V., Matwin, S., de Aguiar Ciferri, C. D., & Ciferri, R. R. (2018). An encryption methodology for enabling the use of data warehouses on the Cloud. International Journal of Data Warehousing and Mining (IJDWM), 14(4), 38-66.

[14] Yadav, S., & Tiwari, V. (2018). Encryption and Obfuscation: Confidentiality technique for enhancing data security in public cloud storage. Journal of Computer and Information Technology, 09, 33-39.

[15] Almeghari, M. J. (2017). Data Warehouse Signature: High performance evaluation for implementing security issues in Data Warehouses through a new framework. Journal of Computer Sciences and Applications, 5(1), 17-24.

[16] dos Santos, R. J. R. (2014). Enhancing data security in Data Warehousing. (Doctoral dissertation in Information Science and Technology). University of Coimbra. Retrieved from http://hdl.handle.net/10316/25230

[17] Vishnu, B., Manjunath, T., & Hamsa, C. (2014). An effective data warehouse security framework. International Journal of Computer Applications, 975, 8887.

[18] Santos, R. J., Rasteiro, D., Bernardino, J., & Vieira, M. (2013). A specific encryption solution for Data Warehouses. Paper presented at the International Conference on Database Systems for Advanced Applications.

[19] Santos, R. J., Vieira, M., & Bernardino, J. (2016). XSX: Lightweight encryption for Data Warehousing environments. Paper presented at the International Conference on Big Data Analytics and Knowledge Discovery.

[20] Singh, A. (2015). Implementation model for access control using log-based security: Practical approach. Paper presented at the 2015 International Conference on Advances in Computer Engineering and Applications.

[21] Achana, R., Hegadi, R. S., & Manjunath, T. (2015). A novel data security framework using E-MOD for big data. Paper presented at the 2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), BUET, Dhaka, Bangladesh.

[22] Rani, R. (2014). Data Warehouse security using log-based analysis: A review. In International Journal of Advanced Research in Computer Science and Software Engineering (Vol. 4, pp. 447-449).

[23] Santos, R. J., Bernardino, J., & Vieira, M. (2011a). Balancing security and performance for enhancing data privacy in Data Warehouses. Paper presented at the 2011IEEE 10th International Conference on Trust,

Security and Privacy in Computing and Communications.

[24] Santos, R. J., Bernardino, J., & Vieira, M. (2011b). A data masking technique for data warehouses. Paper presented at the Proceedings of the 15th Symposium on International Database Engineering & Applications.

[25] Brabson, B. (2004). How to Learn Visual Basic Programming. A step-by-step guide to tweaking your PC experience (There's no secret to writing good code). Maximum PC May 2004, 60-66.

[26] Manu, & Goel, A. (2017). Encryption algorithm using dual modulus. Paper presented at the 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT).

[27] Tongkaw, S., & Tongkaw, A. (2016). A comparison of database performance of MariaDB and MySQL with OLTP workload. Paper presented at the 2016 IEEE Conference on Open Systems (ICOS), Langkawi, Malaysia.

[28] Chandrashekar, P., Dara, S., & Muralidhara, V. (2015). Efficient format preserving encrypted databases. Paper presented at the 2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT).

[29] Sanchez, J. C. (2016). Investigating the star schema benchmark as a replacement for the TPC-H decision support system. Master's Thesis, East Carolina University.

7. APPENDIX A: TPC-H SCHEMA FOR LINEITEM.

The TPC-H consists of eight tables, namely, Supplier, Part, Partsupp, LineItem, Customer, Orders, Nation, and Region as can be seen in Figure 13. The schema represents a simple data warehouse dealing with sales, customers and suppliers. Customers order products, which can be bought from more than one supplier. Every customer and supplier are located in a nation, which in turn is in a geographic region. An order consists of a list of products sold to a customer. The list is stored in LineItem where every row holds information about one order line. There are several date fields both in LineItem and in Orders, which store information regarding the processing of an order (order date, ship date, commit date and receipt date). The central fact table in TPC-H is LineItem although Partsupp can also be considered another fact table.

The Scale Factor (**SF)** determines the ratio at which the data is loaded into a DW database. It is used to increase the size of the database throughout the benchmarking process. The value in front of SF is the number of rows of data for each table.

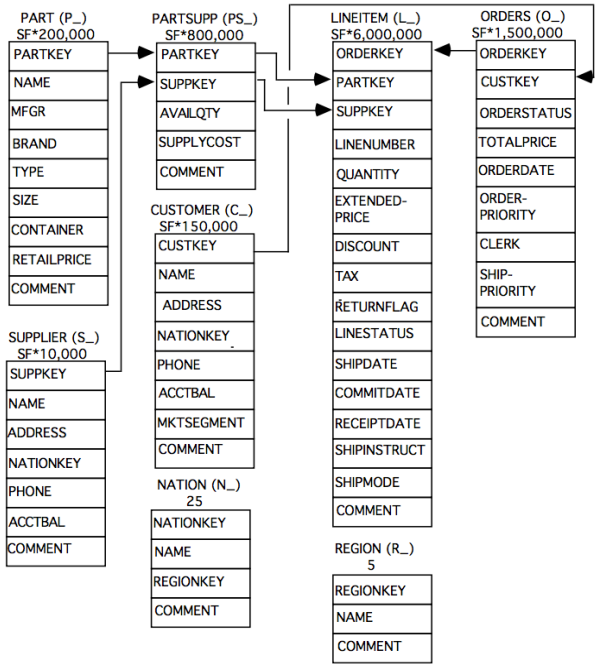

Figure 14. TPC-H Schema. Source: [29]

8. APPENDIX B: DATA MASKING AND ENCRYPTION EXPERIMENTAL RESULTS

In this appendix, we present the averages for the data masking and encryption experimental results. Each result is gotten from the execution of four rounds of experiments and shown in Table 5.

Table 5: Sample Tests run per Test Case

Test 1		
Rows	500,000	
Test case	**Second**	**MB**
Case 1	44.5160	77.6
Case 2	40.8673	77.6
Case 3	44.5714	77.6
Case 4	45.5383	77.6
Average	**44**	**78**

Test 2		
Rows	1,000,000	
Test case	**Second**	**MB**
Case 1	90.3282	148.7
Case 2	91.5370	142.7
Case 3	89.8518	142.7
Case 4	90.1704	143.7
Average	**90**	**144**

Test 3		
Rows	1,500,000	
Test case	**Second**	**MB**
Case 1	134.8201	223.8
Case 2	134.6162	225.8
Case 3	131.4730	224.8
Case 4	153.7349	218.8
Average	**139**	**223**

Test 4		
Rows	3,000,000	
Test case	**Second**	**MB**
Case 1	281.3464	439
Case 2	270.8838	459
Case 3	277.3704	428
Case 4	260.8134	432
Average	**273**	**440**

Encryption Time for the TPC-H 1GB Fact Table per Solution

Test Case	No. of Rows	Data Size (MB)	MOBAT (sec)	MOD95 (sec)	Difference between the Algorithms
Case 1	500,000	78	42	37	5
Case 2	1,000,000	144	87	76	11
Case 3	1,500,000	223	152	134	18
Case 4	3,000,000	440	321	295	26
Tot/Average	**6,000,000**	**885**	**151**	**136**	**15**

Decryption Time for the TPC-H 1GB Fact Table per Solution

Test Case	No. of Rows	Data Size (MB)	MOBAT (sec)	MOD95 (sec)	Difference between the Algorithms
Case 1	500,000	78	34	30	4
Case 2	1,000,000	144	80	79	1
Case 3	1,500,000	223	147	133	14
Case 4	3,000,000	440	282	261	21
Tot/Average	**6,000,000**	**885**	**136**	**126**	**10**