ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Assessing performance of encrypted databases under query processing with the REA Algorithm

Rohini A.Chirde[1]
PRMIT & R
Badnera-Amravati (M.S.)
India

S. S. Kulkarni[2]
Assistant Professor
PRMIT & R
Badnera-Amravati (M.S.)
India

Abstract: ***The security of data is most important in today's world, so that the data should not be accessed by an intruder. The database contains large amount of data that needs to be secure. Encryption is the process of encoding data so that its meaning is not obvious; decryption is reverse process, transforming an encrypted message back into its normal original form. Encryption in database system is an important aspect for research, as secure and efficient algorithms are needed that provide the ability to query over encrypted database and allow optimized encryption and decryption of data. We will discuss about Reverse Encryption Algorithm (REA). The REA is simple and speedy. We examine the query processing performance over encrypted databases with our new encryption algorithm REA and with AES.***

Keywords: Encryption, database security, query processing, encoding, algorithm.

I. INTRODUCTION

Database security has been provided by physical security and operating system security. As far as we know, neither of these methods sufficiently provides a secure support on storing and processing the sensitive data. Encryption in database systems is an important topic for research, as secure and efficient algorithms are needed that provide the ability to query over encrypted database and allow optimized encryption and decryption of data. Clearly, there is a compromise between the degree of security provided by encryption and the efficient querying of the database. This paper provides maximum security and limits the added time cost for encryption and decryption so as to not degrade the performance of a database system.

Database encryption mechanism could provide the following security:

1. Encryption mechanism can prevent users from obtaining data in an unauthorized manner.
2. Encryption mechanism can verify the authentic origin of a data item.
3. Encryption mechanism also prevents from leaking information in a database when storage mediums, such as disks, CD ROM, and tapes, are lost.

However, how to query efficiently the encrypted database becomes a challenge. This usually implies that the system has to sacrifice the performance to obtain the security. When data is stored in the form of cipher, we have to decrypt all the encrypted data before querying them. It is impractical because the cost of decryption over all the encrypted data is very expensive.

So a efficient mechanism is needed which will provide the security as well as provide efficient querying of the data with reduced time cost for encryption and decryption. The innovative encryption algorithm, known as "Reverse Encryption Algorithm (REA)" is efficient and reliable. It has accomplished security requirements and is fast enough for most widely used software. REA encryption algorithm limits the added time cost for encryption and decryption and at the same time improves the performance of the query over encrypted database.

The proposed new encryption scheme (Chaotic Order Preserving Encryption (COPE)) hides the order of the encrypted values by changing the order of buckets in the plaintext domain. It is secure against known plaintext attack. However, COPE can be used just on trusted server where the encryption keys are used to perform many queries such as join and range queries.

The overhead of range queries over encrypted database is much higher than the overhead of range queries over plaintext database. In addition, it uses many keys to change the order of buckets and in some cases that may lead to have duplicated values. Another drawback in COPE is the encryption and decryption cost. That is because of the computation complexity to randomize the buckets and assign the correct order within each bucket.

The bucketing approach is dividing the plaintext domain into many partitions (buckets). The encrypted database in the bucketing approach is augmented with additional information (the index of attributes), thereby allowing query processing to some extent at the server without endangering data privacy. The encrypted database in the bucketing approach contains etuples and corresponding bucket-ids (where many plaintext values are indexed to same bucket-id).

In this scheme, executing a query over the encrypted database is based on the index of attributes. The result of this query is a superset of records containing false positive tuples. These false hits must be removed in a post filtering process after etuples returned by the query are decrypted. Because only the bucket-id is used in a join operation, filtering can be complex, especially when random mapping is used to assign bucket-ids rather than order preserving mapping. In bucketing, the projection operation is not implemented over the encrypted database, because a row level encryption is used.

II. AES(RIJNDAEL)

The Advanced Encryption Standard (AES) was published by NIST (National Institute of Standards and Technology) in 2001. AES is a block symmetric cipher that is intended to replace DES as the approved standard for a wide range of applications. The AES cipher and other candidates forms the latest generation of block ciphers, and now we see a significant increase in the block size – from the old standard of 64-bits up to 128-bits; and keys size of 128, 192, and 256-bits.

NIST selected Rijndael as the proposed AES algorithm. The Evaluation Criteria for selecting AES in the first round are (private key symmetric block cipher, 128-bit data, 128/192/256-bit keys, stronger & faster than Triple-DES, active life of 20-30 years (for long term secrecy

The final criteria for evaluation were general security, ease of software & hardware implementation, implementation attacks, and flexibility (in encrypt/decrypt, keying, and other factors). After testing and evaluation, NIST announced for selection Rijndael as AES. Rijndael algorithm is flexible in supporting any combination of data and key size of 128,192, and 256 bits. However, AES simply allow 128 bit data length that can be divided into four basic operation blocks. These blocks operate on array of bytes and organized as a 4×4 matrix that is called the state. For full encryption, the data is passed through number rounds (10 (key size 128 bits), 12 (key size 192 bits), 14 (key size 256 bits)).

III. THE PROPOSED ENCRYPTION ALGORITHM (REA)

We recommend the new encryption algorithm, "Reverse Encryption Algorithm (REA)", because of its simplicity and efficiency. It can outperform competing algorithms. REA algorithm is limiting the added time cost for encryption and decryption to so as to not degrade the performance of a database system. Our new algorithm (REA) is a symmetric stream cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, making it ideal for securing data.

The REA algorithm encipherment and decipherment consists of the same operations, only the two operations are different:

1) Added the keys to the text in the encipherment and removed the keys from the text in the decipherment.

2) Executed divide operation on the text by 4 in the encipherment and executed multiple operation on the text by 4 in the decipherment.

We execute divide operation by 4 on the text to narrow the range domain of the ASCII code table at converting the text. The details and working of the proposed algorithm REA are given below.

Encryption Algorithm of the REA

We will be presenting the steps of the Reverse Encryption Algorithm REA (Algorithm1).

Step1: Input the text and the key.
Step2: Add the key to the text.
Step3: Convert the previous text to ascii code.
Step4: Convert the previous ascii code to binary data.
Step5: Reverse the previous binary data.
Step6: Gather each 8 bits from the previous binary data and obtain the ascii code from it.
Step7: Divide the previous ascii code by 4.
Step8: Obtain the ascii code of the previous result divide and put it as one character.
Step9: Obtain the remainder of the previous divide and put it as a second character.
Step10: Return encrypted text

Decryption Algorithm of the REA

We will be presenting the steps of the decryption algorithm of the Reverse Encryption Algorithm REA (Algorithm 2).

Step1: Input the encrypted text and the key.
Step2: Loop on the encrypted text to obtain ascii code of characters and add the next character.
Step3: Multiply ascii code of the first character by 4
Step4: Add the next digit (remainder) to the result Of multiplying operation.
Step5: Convert the previous ascii code to binary data.
Step6: Reverse the previous binary data.
Step7: Gather each 8 bits from the previous binary data and obtain the ascii code from it.
Step8: Convert the previous ascii code to text.
Step9: Remove the key from the text.
Step10: Return decrypted data

Figure for Encryption:

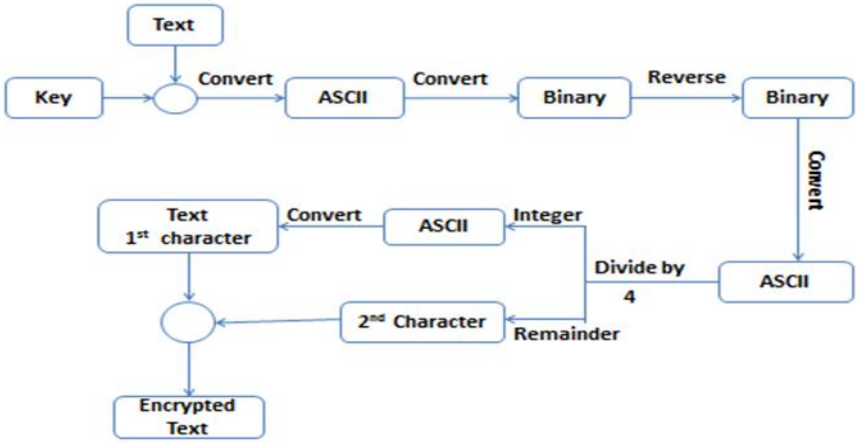

Fig.1 : Steps of the REA encryption algorithm

Figure for Decryption :

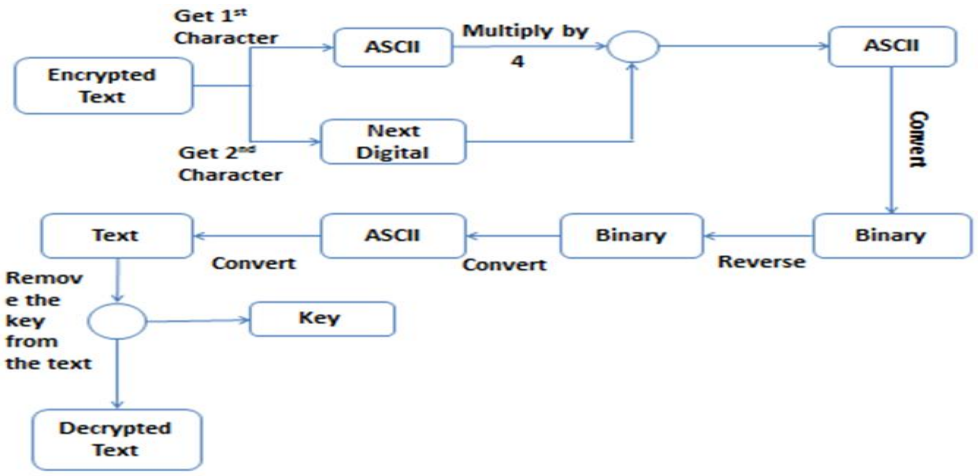

Fig. 2 : Steps of the REA decryption algorithm

REA: An Examples Cipher

We have example on which we have applied our new encryption algorithm REA:

1. Text to explain the running methods the proposed algorithm REA.

Text:

The first example on which we applied our new encryption algorithm REA is on the text, the explanation has been provided below.

The text is: "Welcome! Mousa1976

The key is: "123" (It takes a variable-length key)

Encrypted text is: 323130*0&2$3'0$0$2&27273,2$0"2#0'232122021

Encipherment:

1) Add the key to the text:
123Welcome! Mousa1976.

2) Convert the previous text to ascii code.
1 --> 49, 2 --> 50, 3 --> 51, W --> 87, e -->101, …….

3) Convert the previous ascii code to binary data:
00110001 00110010 00110011 01010111 01100101…..

4) Reverse the previous binary data:
11001110 11001101 11001100 10101000 10011010.….

5) Gather each 8 bits from the previous binary data and obtain the ascii code of it:
206 205 204 168 154.....

6) Divide the previous ascii code by 4 and obtain the ascii of the result(put it as one ascii character) and obtain the remainder (put it as second character).

- 206/4 = 51 ---> 3 and the remainder (next digit) = 2 (put its as 32).

- 205/4 = 51 ---> 3 and the remainder (next digit) = 1 (put its as 31).
- 204/4 = 51 ---> 3 and the remainder (next digit) = 0 (put its as 30).

•168/4 = 42 ---> * and the remainder (next digit) = 0 (put its as *0).

•154/4 = 38 --->& and the remainder (next digit) = 2 (put its as &2).

7) Encrypted text is (see Figure 3):
"323130*0&2$3'0$0$2&27273,2$0"2#0'232122021"

Decipherment:

1) Loop on the encrypted text to get ascii code of characters and add next character.

2) Multiply ascii code of the first character by 4 and add the next digit (remainder):

- The first character = 3 --->ascii code is: 51 and the next digit(remainder)= 2 then new ascii code is: 206 = 51*4+2
- The first character = 3 --->ascii code is: 51 and the next digit(remainder)= 1 then new ascii code is: 205 = 51*4+1
- The first character = 3 --->ascii code is: 51 and the next digit(remainder)= 0 then new ascii code is: 204 = 51*4+0
- The first character = * --->ascii code is: 42 and the next digit(remainder)= 0 then new ascii code is: 168 = 42*4+0
- The first character = & --->ascii code is: 38 and the next digit(remainder)= 2 then new ascii code is: 154 = 38*4+2

3) Convert final ascii code to binary data:

11001110 11001101 11001100 10101000 10011010.…

4) Reverse the previous binary data:

00110001 00110010 00110011 01010111 1100101…..

5) Convert binary data to ascii code and text:
49 50 51 87 101 …..

6) Remove the key from text:
123Welcome! Mousa1976

7) Decrypted text is (see Figure 4):
" Welcome! Mousa1976

IV. PROPOSED SYSTEM IMPLEMENTATION

Requirements and Input Output Specification

Software

- C#.Net, Microsoft SQL 2012

Operating system requirements:

- Windows XP/ Windows 7

Hardware:

- Pentium IV
- Hard Disk 120 GB
- RAM 1GB

V. SYSTEM DESIGN

We focus on the issues that provide maximum security to database at the same time give maximum performance. Database operations are costly in the sense that they take much time for encryption/decryption and querying of the data. So we will discuss here a mechanism that will take less time as well as provide a security to database.

This project has following phases

1. Take the three plaintext database.
2. Keep the first database as it is.
3. Encrypt second database with AES encryption algorithm.
4. Encrypt third database with REA encryption algorithm.
5. Then we apply same no. of queries on the three databases.
6. At the same time we calculate the query execution time for each algorithm
7. Then we will compare the execution times of two algorithms and find the resultant efficient algorithm.
8. We will display the result in terms of tabular and graphical formats.
9. Lastly, we will come to conclusion by obtaining the results.

We will examine query processing performance evaluation over encrypted databases with the proposed algorithm (REA) and with the most common encryption algorithm AES. The performance measure of query processing will be conducted in terms of query execution time.

In the experiments, we use three databases from the database "Northwind" are:

1. Northwind_Plaintext has not any encrypted fields.
2. Northwind _AES has encrypted fields with AES encryption algorithm.
3. Northwind_REA has the same encrypted fields in "Northwind_AES".But, with using our new encryption algorithm REA.

When the encryption of the fields in the databases were completed. The keys are used in the encryption were kept safe in the table encrypted withthe proposed encryption algorithm REA for the database "Northwind_AES" and the database "Northwind_REA"Only the administrator user will get these keys by entering the password in our simulation. After the administrator enters the password and selects the database, one will be able to see the table of the keys encrypted in the databases

Now, we start executing the queries on these databases. Every query from the first to the tenth executes on the database "Northwind_Plaintext" then calculates the execution time and repeats executes on the database "Northwind_AES" then calculates the execution time and repeats the execution again on the database "Northwind_REA" then calculates the execution time

A first point:the database "Northwind_Plaintext" takes less time than the other databases in terms of the query execution time.

A second point: the database "Northwind_AES" is the bigger than other databases in terms of the query execution time. A third point; the database "Northwind_REA" is slower than the database"Northwind_Plaintext" and faster than the database "Northwind_AES" in terms of the query execution time.

Finally, the results showed that the effects the proposed encryption algorithm REA on the database has a very good

performance compared to the algorithm AES. Our new encryption algorithm REA limits the added time cost for encryption and decryption so as to not degrade the performance of the database system, if we compare with other encryption algorithms such as AES encryption algorithm.

The proposed encryption algorithm REA represents a significant improvement over the encrypted databases. Moreover, reducing the overhead of loading on the system for the complexity of the methods that doing decryption and re-encryption the data in the database.

VI. CONCLUSION

We present a novel approach the Reverse Encryption Algorithm (REA) restating its benefits and functions over other similar encryption algorithm. We examine a method for evaluating query processing performance over encrypted database with our new encryption algorithm (REA) and with the most common encryption algorithm AES.

The results show the superiority of the proposed encryption algorithm REA over other encryption algorithm AES with regards to the query execution time. Our new encryption algorithm REA can reduce the cost time of the encryption/decryption operations and improve the performance.

In the future work, we are interested in extending the proposed encryption algorithm REA in order to apply it to other kind of databases such as object oriented DBMSs of the query processing.

References

1. H. Brown, Considerations Database Management System Encryption Security Solution, A Research Report presented to The Department of Computer Science at the University of Cape Town, 2003.
2. S. Castano, M. Fugini, G. Martella, and P. Samarati, Database Security, Addison-Wesley, 1995.
3. A. Ceselli, E. Damiani, S. D. C. D. Vimercati, S.Jajodia, S. Paraboschi, and P. Samarati, “Modeling and assessing inference exposure in encrypted databases,” ACM Transactions on Information System Security, vol. 8, no. 1, pp. 119–152, 2005.
4. J. Daemen and V. Rijmen, “Rijndael: The advanced encryption standard (AES),” Dr. Dobb's Journal, vol. 26, no. 3, pp. 137-139, Mar. 2001.
5. E. Damiani, S. D. C. D. Vimercati, M. Finetti, S. Paraboschi, P. Samarati, and S.Jajodia, “Implementation of a storage mechanism for untrusted dbmss,” IEE Security in Storage Workshop 2003, pp. 38-46, 2003.
6. G. Davida, D. L. Wells, and J. B. Kam, “A database Encryption system with subkeys,” ACM Transactions on Database Systems, vol. 6, no. 2, pp. 312–328, 1981.
7. E. Damiani, S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Metadata management in outsourced encrypted databases,” in The Second VLDB Workshop on Secure Data
8. M. R. Doomun and K. M. S. Soyjaudah, “Analytical comparison of cryptographic techniques for resource-constrained wireless security" International Journal of Network Security, vol. 9, no. 1, pp. 82-94, 2009.
9. D. S. A. Elminaam, H. M. A. Kader, and M. M. Science, Menoufia University, Egypt in 2008. He Hadhoud, “Evaluating the performance of symmetric encryption algorithms," International Journal of Network Security, vol. 10, no. 3, pp. 213-219, 2010

