# Query Processing Performance on Encrypted Databases by Using the REA Algorithm

Ayman Mousa<sup>1</sup>, Elsayed Nigm<sup>2</sup>, Sayed El-Rabaie<sup>3</sup>, Osama Faragallah<sup>3</sup> (Corresponding author: Ayman Mousa)

Department of Computer Science, Workers University, 12311, Tanta, Egypt<sup>1</sup> Department of Mathematics, Zagazig University, 44519, Zagazig, Egypt<sup>2</sup> Department of Computer Science and Engineering, Menoufia University, 32952, Menouf, Egypt<sup>3</sup> (Email: {ayman\_mosa2004, s\_nigm, srabie1, osam\_sal}@yahoo.com) (Received Nov. 16, 2011; revised and accepted Feb. 1, 2012)

# Abstract

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, because the operations of encryption and decryption greatly degrade query performance. In this paper we propose a new encryption algorithm, which we call Reverse Encryption Algorithm (REA). Our new encryption algorithm (REA) is simple and is fast enough for most applications. REA encryption algorithm provides maximum security and limits the added time cost for encryption and decryption to as to not degrade the performance of a database system. Also, we evaluate the query processing performance over encrypted database with our new encryption 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. Results of a set of experiments validate the functionality and usability of the proposed algorithm REA.

Keywords: AES, database security, query processing

# **1** 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. Cryptographic should be used to guide the storage and access of operations and improve the performance. confidential data in a database system. In [1, 7, 12, 14] 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 [18].

For this purpose, we put forward the innovative encryption algorithm, known as "Reverse Encryption Algorithm (REA)". Our new 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. We also provide a thorough description of the proposed encryption algorithm and its processes.

This paper observes a method for evaluating query processing performance over encrypted database with the proposed encryption 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. The experiment results show the advantages of the proposed encryption algorithm REA over other encryption algorithm AES with regards to the query support is another important dimension of database security. execution time. Our new encryption algorithm (REA) can It is complementary to access control and both of them reduce the cost time of the encryption/decryption

> The remainder of this paper is organized as follows. Section 2 discusses related work and overviews of the AES (Rijndael) encryption algorithm. Section 3 describes the proposed encryption algorithm REA. Section 4 shows the simulation results for evaluating query processing performance over encrypted database with the proposed

encryption algorithm REA and other encryption algorithm keying, and other factors). After testing and evaluation, works.

# 2 Related Work

In [16] proposed a new encryption scheme (Chaotic Order Preserving Encryption (COPE)). It 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 [3, 5, 6, 13, 21] is dividing the plaintext domain into many partitions (buckets). The encrypted database in the bucketing approach is augmented that can be effectively used for encryption and safeguarding 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.

#### 2.1 AES (Rijndael): OverviewSubsection

The Advanced Encryption Standard (AES) [4] 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,

AES. Finally Section 5 presents conclusions and future 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  $4 \times 4$  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)).

# **3** 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. In this section we provide a comprehensive yet concise algorithm. We also give a general analysis of the functioning of these structures.

Our new algorithm (REA) is a symmetric stream cipher 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.

## 3.1 Encryption Algorithm of the REA

We will be presenting the steps of the encryption algorithm of the Reverse Encryption Algorithm REA (Algorithm 1). The following steps are (see Figure 1):

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.



Figure 1: Steps of the REA encryption algorithm

#### **Algorithm 1: REA\_Encryption**

INPUT: Plaintext (StrValue), Key (StrKey).

- OUTPUT: Ciphertext (EncryptedData).
- 1: Add the key to Text (StrKey + StrValue)---> full string (StrFullVlaue).
- 2: Convert the Previous Text(StrFullVlaue) to ascii code (hexdata).
- 3: Foreach (byte b in hexdata).
  - a. Convert the Previous ascii code (hexdata) to binary data (StrChar).
    - b. Switch (StrChar.Length).
      - Case 7 ---> StrChar = "0" + StrChar.
      - Case 6 ---> StrChar = "00" + StrChar.
      - Case 5 ---> StrChar = "000" + StrChar.
      - Case 4 ---> StrChar = "0000" + StrChar.
      - Case 3 ---> StrChar = "00000" + StrChar.
      - Case 2 ---> StrChar = "000000" + StrChar.
      - Case 1 ---> StrChar = "0000000" + StrChar.
  - Case 0 ---> StrChar = "00000000" + StrChar.
- c. StrEncrypt += StrChar. (where, StrEncrypt="")
- 4: Reverse the Previous Binary Data(StrEncrypt).
- 5: For i from 0 to StrValue.Length do the following: a. if (binarybyte.Length == 8).
  - i.Convert the binary data (StrEncrypt) to ascii code and, ii.Divide the ascii by  $4 \rightarrow$  the result(first character) and, iii.The remainder of the previous  $\rightarrow$  second character.
- 6: Return (EncryptedData).

## **3.2 Decryption Algorithm of the REA**

We will be presenting the steps of the decryption algorithm of the Reverse Encryption Algorithm REA (Algorithm 2). The following steps are (see Figure 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 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 2: Steps of the REA decryption algorithm

# Algorithm 2: REA\_Decryption

- INPUT: Ciphertext (EncryptedData), the Key (StrKey).
- OUTPUT: Plaintext (DecryptedData),
- 1: For (i = 0; i < EncryptedData.Length; i += 2)
  - a. Get the ascii code of the encrypted text
    - b. newascii = (EncryptedData[i] \* 4) + the next digit(remainder)[i+1].
- 2: Foreach (byte b in newascii).
  - a. Convert the Previous ascii code (newascii) to binary data (StrChar).
    - b. Switch (StrChar.Length).
      - Case 7 ---> StrChar = "0" + StrChar.
      - Case  $6 \rightarrow \text{StrChar} = "00" + \text{StrChar}$ .
      - Case 5 ---> StrChar = "000" + StrChar.
      - Case 4 ---> StrChar = "0000" + StrChar.
      - Case 3 ---> StrChar = "00000" + StrChar.
      - $Case \ 2 \dashrightarrow > StrChar = "000000" + StrChar.$
      - Case 1 ---> StrChar = "0000000" + StrChar.
      - Case 0 ---> StrChar = "00000000" + StrChar.
  - c. StrDecrypt += StrChar.
- 3: Reverse the Previous Binary Data(StrDecrypt).
- 4: For i from 0 to StrDecrypt.Length do the following:
  - a. if (binarybyte.Length == 8).
    i. Convert the binary data (StrChar) to ascii code (hexdata) and,
    - ii. Convert the previous ascii code (hexdata) to the text (StrFullVlaue).
- 5: Remove the key from the text (StrFullVlaue StrKey) → (StrValue).
- 6: Return (DecryptedData).

#### 3.3 REA: An Examples Cipher

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

- 1) *Text* to explain the running methods the proposed algorithm REA.
- Database Microsoft SQL Server 2005 is "Northwind\_Plaintext". The programming tasks were built by Microsoft Visual Studio 2005.net.
- 3.1.1 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:
- í 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).
  - $205/4 = 51 \dots > 3$  and the remainder (next digit) = 1 (put its as 31).
  - $204/4 = 51 \longrightarrow 3$  and the remainder (next digit) = 0 (put its as 30).
  - 168/4 = 42 ---> \* and the remainder (next digit) = 0 (put its as \*0).
  - $154/4 = 38 \dots \&$  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"

Please Enter the Text: Welcome! Mousa1976		-
Please Enter the Key: 123		
Begin Encryption		
Add the Key to Text: 123Welcome! Mousa1976		
Convert the Previous Text to ASCII		
$\begin{array}{cccccccccccccccccccccccccccccccccccc$		
ь> 54 Convert the Previous ASCII to Binary Data		
98118981891198189911891181919118119918911918919191918919911898119191191	010110010 1010011011	1
Reverse the Previous Binary Data		
1 : 961 1 : 81 1 : 961 1 : 91 1 : 961 1 : 961 8 : 81 896 1 : 961 1 : 961 8 : 961 1 : 961 1 : 961 8 : 969 : 969 1 : 961 1 : 91 1 : 1 : 1 : 1 : 1 : 1 : 91 : 9	101001101 101100100	0
Gather each 8 bits and get the ascii of it and divide ascii by 4 and get the ascii of the resultKone character)and get the reminder of the previous and put it as second character		
First Ascii: 206       After Divide:51       New Ascii: 3       Renind         First Ascii: 205       After Divide:51       New Ascii: 3       Renind         First Ascii: 206       After Divide:51       New Ascii: 3       Renind         First Ascii: 168       After Divide:32       New Ascii: 4       Renind         First Ascii: 164       After Divide:38       New Ascii: 5       Renind         First Ascii: 147       After Divide:39       New Ascii: 5       Renind         First Ascii: 147       After Divide:36       New Ascii: 5       Renind         First Ascii: 144       After Divide:36       New Ascii: 5       Renind         First Ascii: 144       After Divide:36       New Ascii: 5       Renind         First Ascii: 144       After Divide:36       New Ascii: 5       Renind         First Ascii: 154       After Divide:36       New Ascii: 5       Renind         First Ascii: 122       After Divide:36       New Ascii: 7       Renind         First Ascii: 122       After Divide:35       New Ascii: 7       Renind         First Ascii: 122       After Divide:35       New Ascii: 7       Renind         First Ascii: 124       After Divide:35       New Ascii: 7       Renind         First Ascii: 144       Aft		
Encrypted Text is: 323130×082\$3'0\$0\$2827273,2\$0"2#0'232122021		

Figure 3: Running the program of the proposed encryption algorithm REA

#### 3.3.2 Database

The second example on which we applied our new encryption algorithm REA is on database Microsoft SQL Server 2005 is called "Northwind Plaintext". The programming tasks were built by Microsoft Visual Studio 2005 .net. In the previous our paper, we encrypted/decrypted some fields from the database "Northwind\_Plaintext" by the proposed encryption algorithm REA and compares with the most common encryption algorithms namely: DES, 3DES, RC2, AES and Blowfish. A comparison has been presented for those encryption algorithms at encryption and decryption time. The experiment results show the superiority of REA algorithm over other algorithms in terms of the encryption and decryption time. We also will use these experiments in this current paper to encrypt some fields of the database "Northwind\_REA" with the proposed encryption algorithm REA (see Figure 6) in Section 4.

Begin Decryption
Loop on the encrypted text to get ascii of characters and add next character multiple Ascii of the first number by 4 and add the next digit(reminder)
First Character Ascii: 51 The Digit(Reminder): 2 New Ascii: 206         First Character Ascii: 51 The Digit(Reminder): 1 New Ascii: 207         First Character Ascii: 42 The Digit(Reminder): 0 New Ascii: 208         First Character Ascii: 42 The Digit(Reminder): 0 New Ascii: 208         First Character Ascii: 42 The Digit(Reminder): 0 New Ascii: 168         First Character Ascii: 36 The Digit(Reminder): 0 New Ascii: 154         First Character Ascii: 36 The Digit(Reminder): 0 New Ascii: 154         First Character Ascii: 36
Convert Final Ascii to Binary Data
110011101100110111001100101010100010011010
Reverse the Previous Binary Data (Correct Binary Data)
00110001001100100001100110101010101010
Convert Binary Data to Text
123Welcome! Mousa1976
Remove the Key from Text: 123Welcome! Mousa1976
Decrypted Text is: Welcome! Mousa1976

Figure 4: Running the program of the proposed decryption algorithm REA

## 4 Simulation Results

A typical case study is studied in this section, to give the 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.

All our experiments were done on laptop IV 2.0 GHz Intel processor with 1 MB cache memory, 1 GB of memory, and one Disk drive 120 GB. The Operating System which was used is Microsoft Windows 7 professional. The simulation results were executed based on the database Microsoft SQL Server 2005 is "Northwind", which contains seven tables. The programming tasks were built by Microsoft Visual Studio 2005 .net.

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 (see example in Figure 5).
- 3) *Northwind\_REA* has the same encrypted fields in "Northwind\_AES". But, with using our new encryption algorithm REA (see example in Figure 6).

Table 1 presented the names of fields are the encrypted in the database *"Northwind\_AES"* and the database *"Northwind\_REA"*.

Table 1: The names of encrypted fields

	Tables	Fields
1	Products	UnitPrice
2	Orders	Freight
3	Order Details	UnitPrice
4	Order Details	Quantity
5	Suppliers	ContactName
6	Employees	Notes
7	Customers	ContactName
8	Customers	ContactTitle







Figure 6: Encrypted the field in the database *"Northwind\_REA"* with the proposed algorithm REA

When the encryption of the eight fields (shown in the table 1) in the databases were completed. The keys are used in the encryption were kept safe in the table encrypted with the proposed encryption algorithm REA for the database "Northwind\_AES" (see Figure 7) and the database "Northwind\_REA" (see Figure 8). Only the administrator user will get these keys by entering the password in our simulation (see Figure 9). 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 "Northwind\_AES" (see Figure 10) and "Northwind\_REA"

# (see Figure 11).

plorer + 7 ×	1	ble - dbo.Key AYM	ANMOUSA_PCsl	nwnd_AES.sql Su	mmary	
• 🛃 = 🖻 Y		StrTable	StrField	StrKey	StrAlgo	StrFieldType
(MANMOUSA PC (SQL Server 9.0.1399 · *	۶.	#3#3\$3%282#1#0	303132/0\$0\$1"3	3031321213123	AES	nvarchar(4000)
Databases		303132,0#1838	303132.1#182	3031323231302	AES	money
🤰 System Databases		303132+3#1\$0	303132*2\$1%2	3031322322212	AES	money
Database Snapshots		303132,0#1838	303132+2"2"2\$1	3031321322212	AES	smallint
Northwind_AES		303132,0#1838	303132*2\$1%2	3031323231302	AES	money
		303132.2\$2#3\$	303132,1\$0"38	3031321212123	AES	ntext
System Tables		303132/0"2#0"3	303132/0\$0\$1"3	3031323232323	AES	nvarchar(4000)
🗉 🔟 dbo.Categories		303132/0"2#0"3	303132/0\$0\$1"3	3031321320121	AES	nvarchar(4000)
🗉 🖬 dbo.CustomerCustomerDer	*	NULL	NULL	NULL	NULL	NULL
🐨 📑 dbo.CustomerDemographic						

Figure 7: Encrypted fields in the keys table "Northwind\_AES" with the proposed algorithm REA

Norer 🗸 🗸 🕽	Ta	ble - dbo.Key AYM	ANMOUSA_PC n:	stnwnd_REA.sql S	ummary	
• 🛃 🖩 🖻 🍸		StrTable	Strield	StrKey	StrAlgo	StrFieldType
MANMOUSA_PC (SQL Server 9.0.1399 - +		#3#3\$3%282#1#0	303132/0\$0\$1"3	3031322221201	REA	nvarchar(4000)
Databases		303132,0#1838	303132.1#182	3031322021222	REA	money
System Databases		303132+3#1\$0	303132*2\$1%2	3031323333121	REA	money
Database Snapshots		303132,0#1838	303132+2*72\$1	3031321323203	REA	smallint
Northwind_AES		303132,0#1838	303132*2\$1%2	3031323231302	REA	money
Northwind RFA		303132.2\$2#3\$	303132,190"38	3031321213123	REA	ntext
Database Diagrams		303132/0"2#0"3	303132/05051*3	3031323231302	REA	nvarchar(4000)
🗄 🫅 Tables		303132/0"2#0"3	303132/05051*3	3031321213202	REA	nvarchar(4000)
	×	NULL	MULL	NULL	NULL	NULL

Figure 8: Encrypted fields in the keys table *"Northwind\_REA"* with the proposed algorithm REA



Figure 9: Login of the administrator to get the keys

Specify Server name	ayma	anmousa_pc		Refresh	
Databases:		Table Name	Field Name	Key	Algorithm
naster		Suppliers	ContactName	98908765434563123112	AES
empdb		Orders	Freight	12345677654321234323	AES
nodel		Products	UnitPrice	45654565456546565565	AES
asdb		Order Details	Quantity	85651111888867767676	AES
Northwind AFS		Order Details	UnitPrice	12345abc323298786fe2	AES
Northwind REA		Employees	Notes	99900088811100022222	AES
		Customers	ContactName	11100023211232319324	AES
		Customers	ContactTitle	87986023211232319131	AES
	*				

Figure 10: Get the keys of the encrypted fields in the database "Northwind\_AES"

Specify Server name	aym	anmousa_pc		Refresh	
Databases:		Table Name	Field Name	Кеу	Algorithm
master	•	Suppliers	ContactName	5678765456787656	REA
tempdb		Orders	Freight	7654323456111000	REA
model		Products	UnitPrice	0098786433432312	REA
msdb		Order Details	Quantity	8472398065572866	REA
Northwind_Plaintext		Order Details	UnitPrice	123498765432cba7	REA
Northwind REA		Employees	Notes	9890123498765432	REA
		Customers	ContactName	1234567891234567	REA
		Customers	ContactTitle	9876543210876543	REA

Figure 11: Get the keys of the encrypted fields in the database "*Northwind\_REA*"

In the experiments, we use ten queries (namely: Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9 and Q10) difference in number the encrypted fields. These queries are varying

when joining the tables. In our simulation, we use three databases are "Northwind\_Plaintext" has no encrypted fields, "Northwind\_AES" has encrypted fields and "Northwind\_REA" has the same encrypted fields in "Northwind\_AES".

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 (see executing query Figure 12) and repeats executes on the database "Northwind\_AES" then calculates the execution time (see executing query Figure 13) and repeats the execution again on the database "Northwind\_REA" then calculates the execution time (see executing query Figure 14).

cify Server n	ame aymanmou	sa_pc		lefresh	
ry (SQL Statem	ent):				
LECT [Categori	ies].[CategoryID],	[Categories].[Cat	egoryName],	[Customers].	Fields
ompanyName],[	Customers].[Coun	try],[Customers].	[CustomerID	].[Order	
ders] [Orderil],	(Order Details).[Q	ight] [Orders] [O	rderID1 (Pro	mcej, ducts]	Relation
ategoryID1.[Proc	ducts].[ProductID]	FROM [Catego	ries].[Custon	uers].[Order	Where & C
tails].[Orders].[I	Products] Where	[Order Details].[	UnitPrice] >	15 and _	- mare ee e
Selec	t DataBase: N	orthwind_Plaintex	t ·	Exec	ute
Selec ry Results:	et DataBase: N	orthwind_Plaintex	t ·	• Exec	orderID
ry Results: CategoryID	t DataBase: No	CompanyName	t	CustomerID	OrderID
ry Results: CategoryID 7	t DataBase: No CategoryName Produce	CompanyName Toms Spezialtäten	Country Germany	CustomerID TOMSP	OrderID 10249
Ty Results: CategoryID 7 2	CategoryName Produce Produce	CompanyName Toms Spezialitäten Hanari Cames	Country Germany Brazil	CustomerID TOMSP HANAR	OrderID 10249 10250
ry Results: CategoryID 7 2 5	CategoryName Produce Produce Contements	CompanyName Toms Spezaltaten Hanari Carnes Hanari Carnes	Country Germany Brazil Brazil	CustomerID TOMSP HANAR HANAR	CrderID 10249 10250 10250
ry Results: CategoryID 7 2 5 5	CategoryName Produce Produce Condments Grains/Greats	CompanyName CompanyName Toms Spezialitäten Hanari Carnes Hanari Carnes Victualitäes en stock	Country Germany Brazil Brazil France	CustomerID TOMSP HANAR HANAR VICTE	Ute OrderID OrderID 10249 10250 10255 1025 102
Velec ry Results: CategoryID 7 2 5 2 -	t DataBase: N CategoryName Produce Produce Condiments Grains/Cereals Condiments	CompanyName CompanyName Toms Spezialitäten Hanari Carnes Hanari Carnes Hanari Carnes Victuallies en stock Victuallies en stock	Country Germany Brazil Brazil France France	CustomerID TCMSP HANAR HANAR VICTE VICTE	Ute OrderID OrderID 10249 10250 10250 10251 10251 10251 10251
rty Results: CategoryID 7 2 5 2 3	category/Name Produce Produce Condiments Condiments Condiments Condiments	CompanyName Tons Spezaltäten Hanari Carnes Victualles en stock Victualles en stock Suprèmes délices	t Country Germany Brazil Brazil France France Belgium	CustomerID TOMSP HANAR HANAR VICTE VICTE SLPRD	CrderID 10249 10250 10250 10251 10251 10251 10251 10252

Figure 12: Executing query on the non-encrypted database *"Northwind\_Plaintext"* 

Table 2 presents the values of the queries execution times (Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9 and Q10) on the three databases ("Northwind\_Plaintext", "Northwind \_AES" and "Northwind\_REA"). Figures 15 and 16 show such comparisons between the values of the queries execution times (Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9 and Q10) on the three databases ("Northwind\_Plaintext", "Northwind\_AES" and "Northwind\_REA").

Spe	cify Serv	er name aym	anmousa_pc				Refresh		
Que	ry (SQL St	atement):							
SE	LECT [Cat	egories].[Catego	oryID],[Cate	gories]	[Category	Nam	e],[Customers].	· 1	Fields
[Ca	ompanyNar	me],[Customers]	[Country],[(	Custon	ers].[Cust	omer.	ID],[Order		
LDel LOe	ansj.[Orae ders] [Cus	tomerID] [Order Det	rs][[Quantu rs][[reight]]	yj,[Or Order	el [Order]]	j (UB DI ID	aencej, vodactel	R	elations
IC:	ategorvID].	[Products].[Prod	ductID1 FRO	OMIC	ategories].	Cust	omers].[Order	When	e & Orde
Det	tails][Orde	rs],[Products]	Where [Ord	er Det	ails].[UnitP	nice]	> 15 and		0 00 0100
	2	Select DataBas	e: Northwi	nd_AI	s		-	Execute	1
	2	Select DataBas	e: Northwi	nd_AI	s		•	Execute	]
	2	Select DataBas	se: Northwi	nd_AI	s		•	Execute	]
Que	t ry Results:	Select DataBas	se: Northwi	ind_Al	S		•	Execute	]
Que	ery Results:	Select DataBas	Quantity	nd_AF	UnitPrice	<u>,</u>	CustomerID1	Execute	] Orde
Que	ery Results: pmerID P	Select DataBas	Quantity 40	nd_AF	UnitPrice 42.4	<b>`</b> .	CustomerID1 TOMSP	Freight 11.61	Crde
Que	ery Results: pmerID P R	Select DataBas	Quantity 40 35	nd_AI	UnitPrice 42.4 42.4	×. ! !	CustomerID1 TOMSP HANAR	Freight 11.61 65.83	Orde 1 1024 1 1025
Qua •	ery Results: pmerID P R R	CerderID 0249 10250 10250	Quantity 40 35 15	nd_Ał	UnitPrice 42.4 42.4 15.8	<b>`</b> . ! ! !	CustomerID1 TOMSP HANAR HANAR	Preight 11.61 65.83 65.83	Orde 1024 1029 1029
Qua •	ery Results: pmerID P R R	Creder1D 10249 10250 10250 10251	Quantity 40 35 15 15	nd_Ał	UnitPrice 42.4 42.4 16.8 15.6	×. ! !	CustomerID1 TOMSP HANAR HANAR VICTE	Freight           11.61           65.83           65.83           41.34	Orde 1029 1025 1025
Qua	ery Results: pmerID P R R R	Creder1D 10249 10250 10250 10251 10251	Quantity 40 35 15 15 20	nd_Al	UnitPrice 42.4 42.4 15.8 15.6 16.8		CustomerID1 TOMSP HANAR HANAR VICTE VICTE	Freight           11.61           65.83           41.34           41.34	Orde 1024 1029 1025 1025 1025
Que	ry Results: pmerID P R R R	Creder1D 10249 10250 10250 10251 10251 10251	Quantity 40 35 15 15 20 40	nd_AI	Lunifrice 42.4 42.4 15.8 15.6 15.6 16.8 64.8	· · · · · · · · · · · · · · · · · · ·	CustomerID1 TOMSP HANAR HANAR VICTE VICTE SLIPRD	Execute Preight 11.61 65.83 65.83 41.34 141.34 51.3	Orde 1024 1029 1025 1025 1025 1025 1025

Figure 13: Executing query on the encrypted database *"Northwind\_AES"* 

Т	Table 2: T	he value	s of the q	ueries ex	ecution t	imes on t	he databa	ases		
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Northwind_Plaintext	0.418	0.282	0.531	4.212	3.119	2.167	1.098	0.978	0.378	3.289
Northwind_AES	2.563	1.279	1.647	19.547	21.029	16.864	5.741	3.822	1.716	19.718
Northwind_REA	1.764	0.854	1.092	16.771	16.118	12.827	4.563	3.167	1.372	15.073



Figure 14: Executing query on the database *"Northwind\_REA"* 



Figure 15: The values of the queries execution times on the databases



Figure 16: The values of the queries execution times on the databases

Our experimental results for these comparisons are shown on Table 2, Figure 15 and Figure 16 at the query processing performance (cost time) over encrypted databases. А first point; database the "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 databases.

# 5 Conclusions and Future Work

There is a lot of very important data in the database, which need to be protected from attack. Cryptographic support is an important mechanism of securing them. People, however, must tradeoff performance to ensure the security because the operation of encryption and decryption greatly degrades query performance. For the query types that require extra query processing over encrypted database, the cost differentials of query processing between nonencrypted and encrypted database increase linearly in the size of relations. To solve such a problem, the proposed encryption algorithm REA can implement SQL query over the encrypted database.

In this paper, we will introduce a new encryption algorithm, which we call "Reverse Encryption Algorithm (REA)", restating its benefits and functions over other similar encryption algorithms. REA algorithm limits the added time cost for encryption and decryption so as to not degrade the performance of a database system. We also provide a thorough description of the proposed algorithm and its processes. This paper examines 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 performance measure of query processing will be conducted in terms of query execution time. The results of a set of experiments 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. [12]

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 distributed DBMSs and [13] object oriented DBMSs of the query processing.

# References

- [1] H. Brown, *Considerations in Implementing A 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] 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 Management, Lecture Notes in Computer Science*, pp. 16–32, Springer, 2005.
- [7] 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.
- [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. Hadhoud, "Evaluating the performance of symmetric encryption algorithms," *International Journal of Network Security*, vol. 10, no. 3, pp. 213-219, 2010.
- [10] D. S. A. Elminaam, H. M. A. Kader, and M. M. Hadhoud, "Evaluating the effects of symmetric cryptography algorithms on power consumption for different data types" *International Journal of*

Network Security, vol. 11, no. 2, pp. 78-87, Sep. 2010.

- [11] C. Gu and Y. Zhu, "New efficient searchable encryption schemes from bilinear pairings," *International Journal of Network Security*, vol. 10, no. 1, pp. 25-31, Jan. 2010.
- [12] H. Hacig¨um¨us, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proceedings of ICDE*, pp. 29–38, 2002.
- [13] H. Hacig¨um¨us, B. R. Iyer, and S. Mehrotra, "Ensuring the integrity of encrypted databases in the database-as-a-service model," *DBSec 17th Annual Working Conference on Data and Application Security*, pp. 61–74, Kluwer, 2003.
- [14] J. He and M. Wang, "Cryptography and relational database management system," *IDEAS*, pp. 273–284, 2001.
- [15] H. Kadhem, T. Amagasa, and H. Kitagawa, "Mvopes: Multivalued-order preserving encryption scheme: A novel scheme for encrypting integer value to many different values," *IEICE Transactions*, vol. 93-D, no. 9, pp. 2520–2533, 2010.
- [16] S. Lee, T. Park, D. Lee, T. Nam, and S. Kim, "Chaotic order preserving encryption for efficient and secure queries on databases," *IEICE Transactions on Information and Systems*, vol. 92, pp. 207–217, 2009.
- [17] H. H. Ngo, X. Wu, P. D. Le, C. Wilson, and B. Srinivasan, "Dynamic key cryptography and applications" *International Journal of Network Security*, vol. 10, no. 3, pp. 161-174, May 2010.
- [18] Oracle, Oracle9i Database Security for eBusiness, An Oracle White Paper, June 2001.
- [19] B. Schneier, Applied Cryptography Second Edition: Protocols, Algorithms, and Source, Beijing: China Machine Press, 2000.
- [20] W. Stallings, Cryptography and Network Security Principles and Practice, 4<sup>th</sup> Ed. Prentice-Hill Inc., 2005.
- [21] Q. Tang and D. Ji "Verifiable attribute based encryption," *International Journal of Network Security*, vol. 10, no. 2, pp. 114-120, Mar. 2010.

Ayman Mousa was born on May 11, 1976 in Shebin El-Kom, Menoufia, Egypt. He obtained the B.S. in Math and Computer Science from Faculty of Science, Menoufia University, Egypt in 1998. He obtained his M.Sc. degree in Computer Science also from Faculty of Science, Menoufia University, Egypt in 2008. He is currently a Lecturer of Computer Science in Workers University since 2001. He majors in Database Security and Cryptography.

**Elsayed Nigm**, Professor of Mathematics Department of Mathematics, Faculty of Science, Zagazig University, Zagazig, Egypt. He obtained the B.Sc. in Mathematics, Statistics and computer Sciences, Zagazig University, Faculty of Science, Egypt (1983). He obtained the M.Sc. degree in Mathematics (functional analysis), Zagazig University, Faculty of Science, Egypt (1987). He obtained his Ph.D. degree Mathematics (Mathematical Statistics) also from Zagazig University, Faculty of Science, Egypt (1990). He honors awards prize of the National committee of Mathematics, by Egyptian Academy of Sciences and Technologies, Egypt (2000). Prof E. M. Nigm has published more than 51 papers in international journals, international conferences, local journals and local conferences.

Sayed El-Rabaie (Senior Member, IEEE'1992-MIEE-Chartered Electrical Engineer) Was Born in Sires Elian (Menoufia), EGYPT in 1953. He Received the BSc with Honors in radio communications from Tanta University, Egypt, 1976, MSc in communication systems from Menoufia University, Egypt, 1981, and a PhD in microwave device engineering from the Queen's University of Belfast in 1986. He was a postdoctoral fellow in the Queen's University Department of Electronic Engineering until 1989. In 1992, he was a Research Fellow at the North Arizona University, College of Engineering and Technology, and in 1994 he served as a visiting professor at Ecole Polytechnique de Montreal, Quebec, Canada. Prof. El-Rabaie has authored and coauthored more than 70 papers and technical reports, and 15 books. In 1993, he was awarded the Egyptian Academic Scientific Research Award (Salah Amer Award of Electronics), and in 1995, he received the Award of Best Researcher on CAD from Menoufia University. He is now the vice dean of postgraduate studies and research, Faculty of Electronic Engineering, Menoufia University.

**Osama Faragallah** received his BS in 1997, MSc in 2002, and PhD in 2007, all in computer science and engineering, from Menoufia University, Faculty of Electronic Engineering, Egypt. He was a demonstrator at the Department of Computer Science and Engineering, at Menoufia University, from 1997 to 2002, became an assistant lecturer in 2002, and was promoted to a lecturer in 2007. His research interests cover computer networks, network security, cryptography, Internet security, multimedia security, image encryption, watermarking, steganography, data hiding, and chaos theory.