Evaluating the Performance of Reverse Encryption Algorithm (REA) on the Databases

Ayman Mousa[1], Osama Faragallah[2], Elsayed Nigm[3], and Elsayed Rabaie[2]
[1]Department of Computer Science, Workers University, Egypt
[2]Department of Computer Science and Engineering, Menoufia University, Egypt
[3]Department of Mathematics, Zagazig University, Egypt

Abstract: *Database encryption is a well established technology for protecting sensitive data. Unfortunately, the integration of existing encryption techniques with database systems causes undesirable performance degradation. It is a crucial technique in the security mechanisms of database. In this paper we propose a new encryption algorithm, which we call Reverse Encryption Algorithm (REA). Our new encryption algorithm REA is simple and yet leads to a cipher. It has achieved security requirements and is fast enough for most applications. REA algorithm is limiting the added time cost for encryption and decryption to don't degrade the performance of a database system. Also, we evaluate the performance of the proposed encryption algorithm REA and compare with the most common encryption algorithms. The performance measure of encryption schemes will be conducted in terms of encryption / decryption time. Experiment results show that our new algorithm outperforms other algorithms at encryption and decryption time.*

Keywords: *Database security, cryptographic algorithms, database encryption.*

Received February 2, 2012; accepted June 2, 2013; published online August 5, 2012

1. Introduction

In most organizations, databases hold a critical concentration of sensitive information, and as a result, databases are vulnerable, therefore database system should be protected from any attacks. Today, enhancing the security of a database is becoming one of the most urgent tasks in database research and industry. Thus, many organizations cannot work properly if their database is down; they are normally referred to as mission critical system. Along with the wide application of database comes the need for its protection. Universally, huge amount of effort, time and resources are been spent in trying to make database systems meet security requirements normally include [2]: 1). Prevention of unauthorized disclosure and modification of information. 2). Prevent denial of service. 3). Prevent system penetration by unauthorized person. 4). Prevent the abuse of special privileges.

Designing a database that will achieve these security requirements is very difficult, since a database system processes large amount of data in complex ways. The result is that most conventional database systems have leaks that an attacker can use to penetrate the database. No matter what degree of security is put in place, sensitive data in databases are still vulnerable to attack. A remedy therefore is to turn to cryptographic means of storing data. Encrypting data stored in a database can prevent their disclosure to attackers even if they manage to circumvent the access control mechanism. Database encryption is widely adopted to ensure data privacy, which can prevent attacks from both outside intruders and inside malicious users [16].

We put forward the innovative encryption algorithm, known as Reverse Encryption Algorithm (REA). The proposed encryption algorithm REA is efficient and reliable. It has accomplished security requirements and is fast enough for most widely used software. REA algorithm limits the added time cost for encryption and decryption and at the same time does not degrade the performance of a database system. We also, provide a thorough description of the proposed algorithm and its processes.

This paper observes a method for evaluating performance of our new 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 results show the advantages of the proposed encryption algorithm REA over other encryption algorithms with regards to the encryption and decryption time. The remainder of this paper is organized as follows: Section 2 discusses related work about the performance of the encryption algorithms. Section 3 describes the proposed encryption algorithm REA. Section 4 shows the simulation results for the performance evaluation of our new encryption algorithm REA and compares it with the most common encryption algorithms. Finally, section 5 presents conclusions and future work.

2. Related Work

To give more prospective about the performance of the compared algorithms, this section discusses the results obtained from other resources. It was concluded in [7] that AES is faster and more efficient than other encryption algorithms. When the transmission of data is considered there is an insignificant difference in the performance of different symmetric key schemes (most of the resources are consumed for data transmission rather than computation). Even under the scenario of data transfer it would be advisable to use AES scheme in case the encrypted data is stored at the other end and decrypted multiple times.

A study in [12], is conducted for different popular secret key algorithms such as DES, 3DES, AES, and Blowfish. They were implemented, and their performance was compared by encrypting input files of varying contents and sizes. The algorithms were tested on two different hardware platforms, to compare their performance. They had conducted it on two different machines: PII 266MHz and P4 2.4GHz. The results showed that Blowfish had a very good performance compared to other algorithms. Also, it showed that AES had a better performance than 3DES and DES. It also, shows that 3DES has almost 1/3 throughput of DES, or in other words it needs 3 times than DES to process the same amount of data [3].

3. The Proposed Encryption Algorithm REA

We recommend the new 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 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 operations 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 REA Algorithm 1. The following steps are as shown in Figure 1:

- *Step 1:* Input the text and the key.
- *Step 2:* Add the key to the text.
- *Step 3:* Convert the previous text to ASCII code.
- *Step 4:* Convert the previous ASCII code to binarydata.
- *Step 5:* Reverse the previous binary data.
- *Step 6:* Gather each 8 bits from the previous binary data and obtain the ascii code from it.
- *Step 7:* Divide the previous ascii code by 4.
- *Step 8:* Obtain the ascii code of the previous result divide and put it as one character.
- *Step 9:* Obtain the remainder of the previous divide and put it as a second character.
- *Step 10:* Return encrypted text.

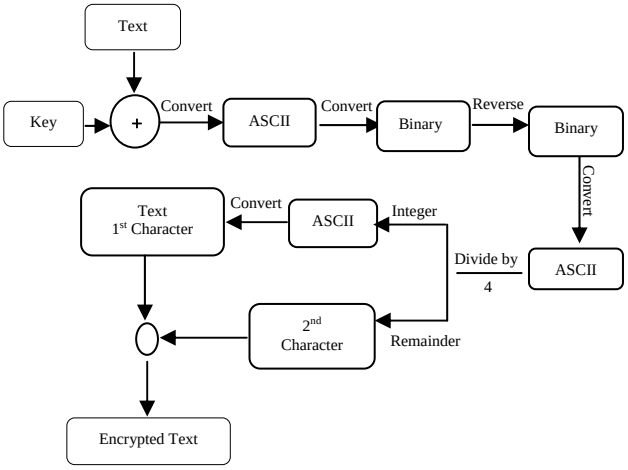

Figure 1. Steps of the REA encryption algorithm.

Algorithm 1. REA-encryption algorithm

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).*
   - 3.1. *Convert the Previous ascii code (hexdata) to binary data (StrChar).*
   - 3.2. *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.
   - 3.3. *StrEncrypt += StrChar. (where, StrEncrypt= "")*
4. *Reverse the Previous Binary Data(StrEncrypt).*
5. *For i from 0 to StrValue.Length do the following:*
   - 5.1. *if (binarybyte.Length == 8).*

5.1.1. Convert the binary data (StrEncrypt) to ascii code and,
5.1.2. Divide the ascii by 4 → the result(first character) and,
5.1.3. The remainder of the previous → second character.

6. Return (EncryptedData).

3.2. Decryption Algorithm of the REA

We will be presenting the steps of the decryption algorithm of the REA Algorithm 2. The following steps are as shows in Figure 2:

- *Step 1:* Input the encrypted text and the key.
- *Step 2:* Loop on the encrypted text to obtain ASCII code of characters and add the next character.
- *Step 3:* Multiply ascii code of the first character by 4.
- *Step 4:* Add the next digit (remainder) to the result multiplying operation.
- *Step 5:* Convert the previous ascii code to binary data.
- *Step 6:* Reverse the previous binary data.
- *Step 7:* Gather each 8bits from the previous binary data and obtain the ascii code from it.
- *Step 8:* Convert the previous ascii code to text.
- *Step 9:* Remove the key from the text.
- *Step 10:* Return decrypted data.

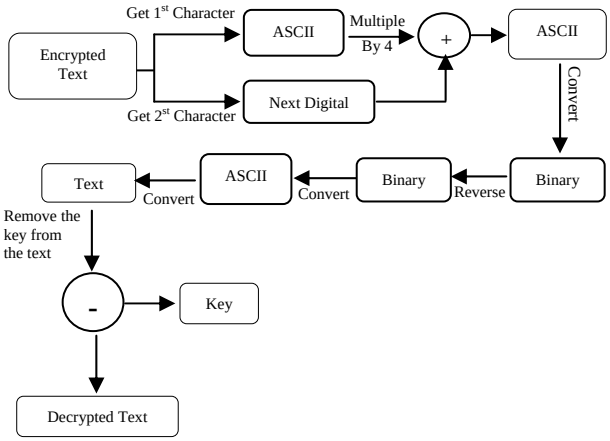

Figure 2. Steps of the REA decryption algorithm.

Algorithm 2. REA-decryption algorithm

Input: Ciphertext (EncryptedData), the Key (StrKey).
Output: Plaintext (DecryptedData),

1. For (i = 0; i < EncryptedData.Length; i += 2)
1.1. Get the ascii code of the encrypted text
*1.2. newascii=(EncryptedData[i]*4)+the next digit(remainder)[i+1].*
2. Foreach (byte b in newascii).
1.1. Convert the Previous ascii code (newascii) to binary data (StrChar).
1.2. 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.
1.3. StrDecrypt += StrChar.
3. Reverse the Previous Binary Data(StrDecrypt).
4. For i from 0 to StrDecrypt.Length do the following:
4.1. if (binarybyte.Length == 8).
4.1.1. Convert the binary data (StrChar) to ascii code (hexdata) and,
4.1.2. 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 on the text and database.

3.3.1. 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)

3.3.1.1. 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---> 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 (nextdigit)=2 (put its as &2).
   - *Etc.*
7. Encrypted text is as shows Figure 3: 323130*0&2$3'0$0$2&27273,2$0"2#0'232122021

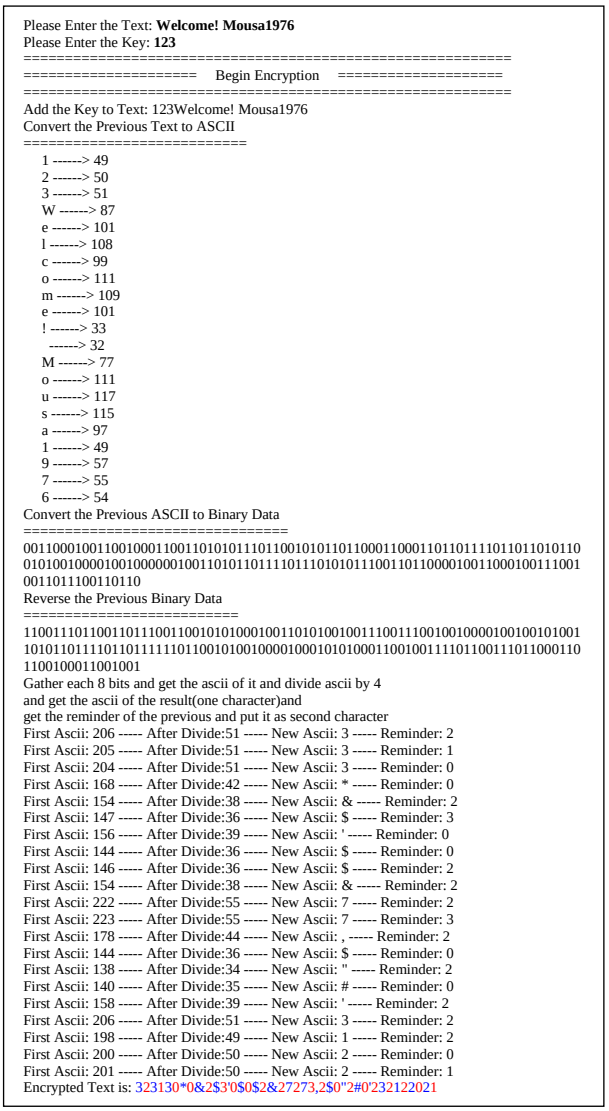

Figure 3. Running the program of the proposed encryption algorithm REA.

3.3.1.2. 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
   - *Etc.*
3. Convert final ascii code to binary data: 11001110 11001101 11001100 10101000 10011010 ….
4. Reverse the previous binary data: 00110001 00110010 00110011 01010111 01100101 ….
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: "Welcome! Mousa1976" as shows in Figure 4.

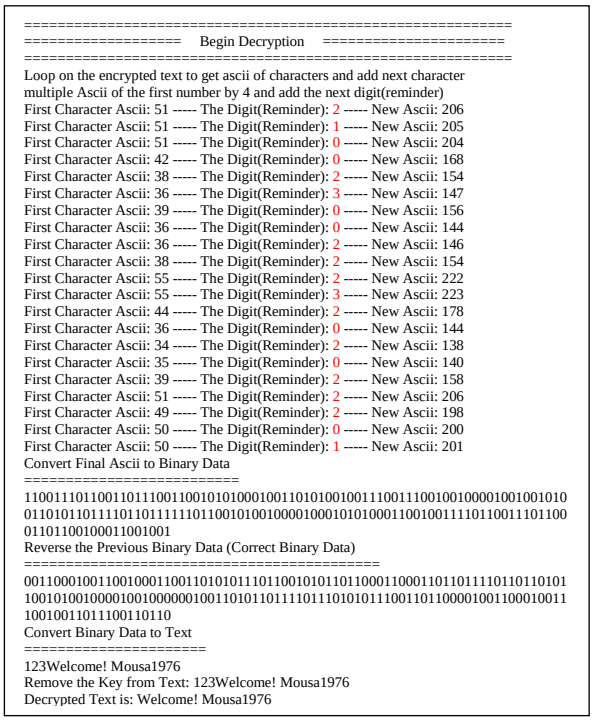

Figure 4. Running the program of the proposed decryption algorithm REA.

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.

We encrypted some fields from the database "Northwind_Plaintext" by the most common encryption algorithms namely: DES, 3DES, RC2, AES, Blowfish and our new algorithm REA. These encryption and decryption achieved by our simulation is there exist in the section 4.

The keys are used in the encryption kept safe in the table encrypted by our new algorithm REA as shown in Figure 5. Only the administrator user will get these keys by entering the password as shown in Figure 6. After the administrator enters the password and selects the database "Northwind_Plaintext" only then, will the administrator be able to see the table of the keys encrypted in the database "Northwind_Plaintext" from Microsoft SQL Server as shown in Figure 7.

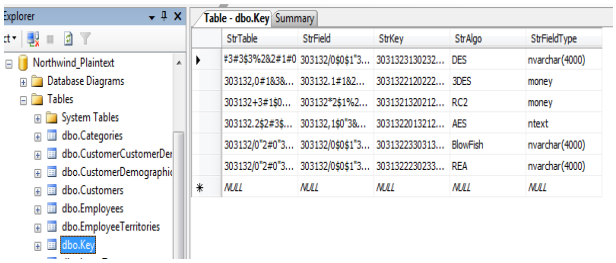

Figure 5. Encrypted fields in the keys table with the proposed algorithm REA.

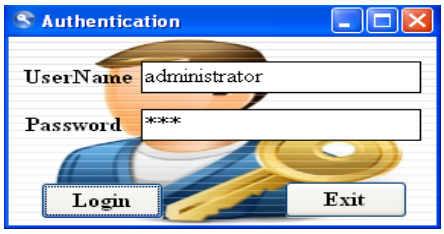

Figure 6. Login of the administrator to get the keys.

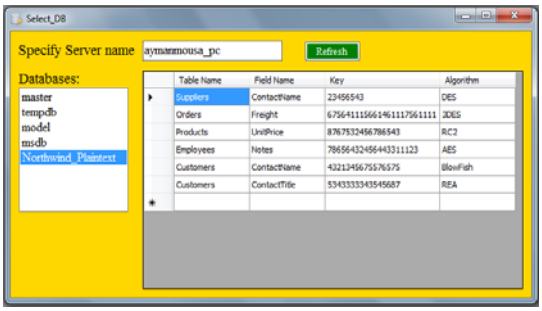

Figure 7. Keys table is decrypted with the proposed algorithm REA.

4. Simulation Results

A typical case study is studied in this section, to give the performance evaluation of a new algorithm REA and to compare it with the most common encryption algorithms namely: DES, 3DES, RC2, AES and Blowfish.

A comparison has been conducted for those encryption algorithms at encryption and decryption time. The encryption time is considered the time that an encryption algorithm takes to produce a ciphertext from plaintext. It indicates the speed of encryption. The decryption time is considered the time that decryption algorithm takes to produce a plaintext from ciphertext. Also, it indicates the speed of decryption.

All our experiments were done on laptop IV 2.0GHz Intel processor with 1MB cache memory, 1GB of memory, and one Disk drive 120GB. 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_Plaintext", which contains seven tables. The programming tasks were built by Microsoft Visual Studio 2005.net. In the experiments, the laptop encrypts a different field size ranges from 77 rows to 2155 rows from different tables in the database "Northwind_Plaintext" see in Table 1. The performance metrics of the encryption time and decryption time have been collected below.

We encrypted ten different fields shown in Table 1 with the proposed encryption algorithm REA and calculated elapsed time for each one. Then, we calculated the averages of the elapsed times shown in Table 2. We repeated this step on other encryption algorithms namely: DES, 3DES, RC2, AES and Blowfish. The Figure 8 has shown one step from ten of our new encryption algorithm REA.

Table 1. The names fields for encrypted and decrypted.

	Table Name	Field Name
1	Suppliers	ContactName
2	Orders	Freight
3	Orders	ShipName
4	Products	UnitPrice
5	Products	QuantityPerUnit
6	Order Details	UnitPrice
7	Order Details	Quantity
8	LargeText	Text
9	Customers	ContactName
10	Customers	ContactTitle

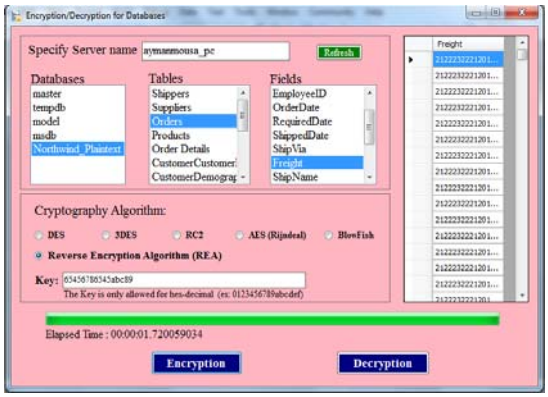

Figure 8. Encrypted the field with the proposed encryption algorithm REA.

We decrypted the same ten different fields shown in Table 1 with our new decryption algorithm REA and calculated elapsed time for each one. Then, we calculated the averages of the elapsed times shown in Table 3. We repeated this step on other decryption algorithms namely: DES, 3DES, RC2, AES and Blowfish. The Figure 9 has shown one step from ten of our new decryption algorithm REA.

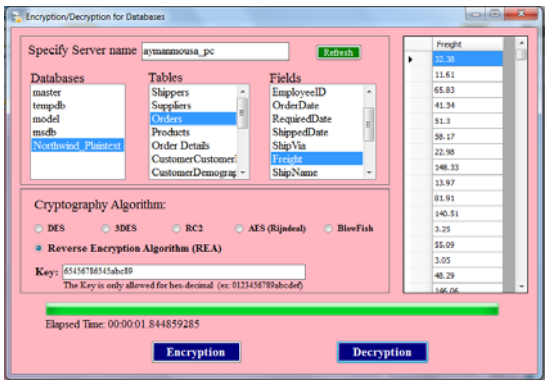

Figure 9. Decrypted the field with the proposed decryption algorithm REA.

The results for this comparison are shown on Table 2 and Figure 10 at the encryption time and Table 3 and Figure 11 at the decryption time. A first point; the results show the superiority of REA algorithm over other algorithms in terms of the encryption and decryption time. A second point; that Blowfish requires less encryption and decryption time than all algorithms except REA. A third point; that AES has an advantage over other 3DES, DES and RC2. A fourth point; that 3DES has low performance in terms of encryption and decryption time when compared with DES. It requires always more time than DES because of its triple phase encryption characteristics. A final point; it is found that RC2 has low performance in terms of encryption and decryption time when compared with other five algorithms.

Table 2. Comparative elapsed times (milliseconds) of encryption algorithms.

	DES	3DES	RC2	AES	BF	REA
1	0.141	0.263	0.342	0.109	0.116	0.104
2	4.063	4.469	4.513	3.297	2.544	1.720
3	3.609	4.484	4.594	3.047	2.671	2.521
4	0.359	0.419	0.395	0.329	0.296	0.266
5	0.344	0.453	0.449	0.331	0.274	0.265
6	14.194	14.968	15.234	14.000	11.304	11.297
7	15.906	17.547	17.328	15.484	12.452	12.360
8	2.960	3.203	3.531	2.688	2.051	2.005
9	0.422	0.463	0.487	0.403	0.376	0.335
10	0.421	0.438	0.442	0.386	0.346	0.334
Avg. Time	**4.2419**	**4.6707**	**4.7315**	**4.0074**	**3.2430**	**3.1207**

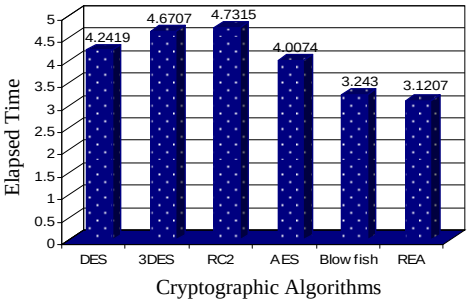

Figure 10. Averages elapsed times of each encryption algorithm.

Overall, the results showed that REA has a very good performance compared to other algorithms. Also, it showed that Blowfish and AES have a better performance than DES, 3DES, and RC2.

Table 3. Comparative elapsed times (milliseconds) of decryption algorithms.

	DES	3DES	RC2	AES	BF	REA
1	0.125	0.141	0.156	0.135	0.137	0.121
2	4.313	4.625	4.516	4.103	3.417	1.845
3	4.672	4.992	5.172	4.212	3.816	3.445
4	0.343	0.359	0.384	0.344	0.322	0.271
5	0.359	0.404	0.426	0.359	0.318	0.281
6	16.687	19.156	20.281	14.266	12.963	12.687
7	17.797	20.313	21.406	15.125	13.761	11.030
8	3.312	3.891	3.906	3.319	2.175	2.743
9	0.443	0.478	0.499	0.421	0.381	0.318
10	0.438	0.447	0.456	0.398	0.315	0.308
Avg. Time	4.8489	5.4806	5.7202	4.2682	3.7605	3.3049

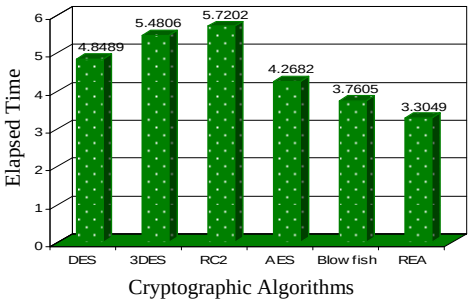

Figure 11. Averages elapsed times of each decryption algorithm.

5. Conclusions and Future Work

It is necessary to consider the evident discrimination between the cryptograph for database security and the traditional cryptograph security. Encrypting sensitive data in the database becomes more and more crucial for protecting from being misused by intruders who bypass conventional access control mechanisms and have direct access to the database files. For this, we propose, in this paper to address this issue and contribute the following.

First, we will introduce a new encryption algorithm, which we call REA, restating its benefits and functions over other similar encryption algorithms. REA algorithm is limiting the added time cost for encryption and decryption so as to not degrade the performance of a database system.

Second, it examines a method for evaluating the performance of our new encryption algorithm REA and compares it with the most common encryption algorithms namely: DES, 3DES, RC2, AES and Blowfish. A comparison has been conducted for those encryption algorithms at encryption and decryption time. The encryption time is considered the time that an encryption algorithm takes to produce a ciphertext from plaintext. It indicates the speed of encryption. The decryption time is considered the time that decryption algorithm takes to produce a plaintext from ciphertext. Also, it indicates the speed of decryption. The results show the superiority of REA algorithm over other algorithms in terms of the encryption and decryption time.

In the future work, we are interested in extending our new encryption algorithm REA to support query processing performance on encrypted database. In addition, we would like to extend and apply our new encryption algorithm REA in other kind of databases such as distributed DBMSs and object oriented DBMSs.

References

[1] Bouganim L. and Pucheral P., "Chip-Secured Data Access: Confidential Data on Untrusted Servers," *in Proceedings of the 28th International Conference on Very Large Data Bases*, China, pp. 131-142, 2002.

[2] Castano S., Fugini M., Martella G., and Samarati P., *Database Security*, Addison-Wesley, USA, 1995.

[3] Chen G., Chen K., and Dong J., "A Database Encryption Scheme for Enhanced Security and Easy Sharing," *in Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*, Nanjing pp. 1-6, 2006.

[4] Coppersmith D., "The Data Encryption Standard and Its Strength Against Attacks," *IBM Journal of Research and Development*, vol. 38, no. 3, pp. 243-250, 1994.

[5] Daemen J. and Rijmen V., "Rijndael: The Advanced Encryption Standard," *Dr. Dobb's Journal*, pp.137-139, 2001.

[6] Damiani E., Vimercati S., and Foresti S., "Key Management for Multi-User Encrypted Databases," *in Proceedings of ACM Storage*, Italy, pp. 74-83, 2005.

[7] El-Fishawy N., "Quality of Encryption Measurement of Bitmap Images with RC6, MRC6, and Rijndael Block Cipher Algorithms," *International Journal of Network Security*, vol. 5, no. 3 pp. 241-251, 2007.

[8] Ferraiolo D. and Kuhn R., "Role-Based Access Controls," *in Proceedings of NIST-NCSC*, Baltim, pp. 554-563, 2002.

[9] Jingmin H. and Wang M., "Cryptography and Relational Database Management Systems," *in Proceedings of IEEE Symposium on the International Database Engineering and Applications*, USA, pp. 273-284, 2001.

[10] Kim Y. and Hong E., "A Study of UniSQL Encryption System: Case Study of Developing SAMS," *in Proceedings of the 9th International Conference on Advanced Communication Technology*, Gangwon, vol. 1, pp. 577-582, 2007.

[11] Mattsson U., "A Database Encryption Solution that is Protecting Against External and Internal Threats, and Meeting Regulatory Requirements: A Practical Implementation of Field Level Privacy," *in Proceedings of the 7th IEEE International Conference on E-Commerce Technology*, USA, pp. 559–565, 2005.

[12] Salama D., Abdual-Kader H., and Hadhoud M., "Studying the Effects of Most Common Encryption Algorithms," *International Arab Journal of e-Technology*, vol. 2, no. 1, pp. 1-10 2011.

[13] Salama D., Abdual-Kader H., and Hadhoud M., "Wireless Network Security Still Has no Clothes," *International Arab Journal of e-Technology*, vol. 2, no. 2, pp. 112-123, 2011.

[14] Schneier B., *Applied Cryptography Second Edition: Protocols, Algorithms, and Source*, China Machine Press, Beijing, 2000.

[15] Schneier B., "The Blowfish Encryption Algorithm," available at: http://pocketbrief.net /related/BlowfishEncryption.pdf, last visited: 2008.

[16] Stallings W., *Cryptography and Network Security Principles and Practice*, Prentice-Hill, 2005.

Ayman Mousa obtained his BS in math and computer science from Faculty of Science, Menoufia University, Egypt in 1998. He obtained his MSc degree in computer science also, from Faculty of Science, Menoufia University, Egypt in 2008 and submitted for PhD from November 2009 in Faculty of Science, Zagazig University. He is currently a lecturer of computer science in Workers University since 2001. His research interests in database security and cryptography.

Osama Faragallah received his BSc 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 include computer networks, network security, cryptography, internet security, multimedia security, image encryption, watermarking, steganography, data hiding, and chaos theory.

Elsayed Nigm is a professor of mathematics, Department of Mathematics, Zagazig University, Egypt. He obtained his BSc in mathematics, statistics and computer sciences, Zagazig University, Faculty of Science, Egypt. He obtained his MSc degree in mathematics (functional analysis), Zagazig University, Faculty of Science, Egypt. He obtained his PhD degree mathematical statistics also, from Zagazig University, Faculty of Science, Egypt in 1990. He honors awards prize of the National Committee of Mathematics, by Egyptian Academy of Sciences and Technologies, Egypt in 2000. He has published more than 51 papers in international journals, international conferences, local journals and local conferences.

Elsayed Rabaie received his 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. Is a senior Member, IEEE'1992- MIEEChartered Electrical Engineer. He was a postdoctoral fellow in the Queen's University at the 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, Canada. He 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.