Safety Analysis of Reverse Algorithm Encryption in Databases

Branko Latinović[1], Zoran Ž. Avramović[1], Mahir Zajmović[2]

[1]PanEuropien University APEIRON, Banja Luka, B&H

[2]PhD Student at University "VITEZ", Vitez, B&H

General Survey

DOI: **10.7251/JIT1901029L** UDC: **004.7.056.55:004.432.2**

Abstract: Encryption provides security for databases. This paper provides a new encryption algorithm ,"Reverse Encryption Algorithm (REA)". Furthermore, designing a REA algorithm has improved data encryption security. Safe and successful proposed encryption algorithm REA is evaluated and compared with the most common encryption algorithms. The designing of the REA algorithm also improves the security of data encryption. Additionally, the safety and the performance of the suggested encryption algorithm REA represents evaluation and enhancement with the most common encryption algorithms. Experimental results show that the proposed encryption algorithm REA surpasses other encryption algorithms in performance and security of databases. All in all, the proposed encryption algorithm REA achieves a balance between security and efficiency.

Keywords: encryption, cryptography, databases, algorithm, REA.

Introduction

Cryptography is the science of data encryption. Cryptographs create algorithms which use input data, called plain text and convert it into encrypted output. Encryption is much more than just moving letters or changing some letters. After a suitable cryptographic encryption, the output is indistinguishable or output looks like a random order of data. For data protection in electronic form new protocols are needed to ensure secrecy as much as the old protocols. There we have a unique opportunity for progress and increased security [7].

Database encryption is a well-established technology for sensitive data protection. Unfortunately, the integration of existing encryption techniques with database systems causes undesirable performance degradation. This is a main technique in safety mechanisms of databases. The database encryption solution is specialized and complex although the internal resources do not have cryptographic expertise in relation to the database environment, external expertise should be used to ensure superior efficiency and strong safety. [1]

A new innovative encryption algorithm REA has been proposed. The algorithm is efficient and secure. It has achieved the safety and is fast enough for the most used software. The proposed algorithm REA limits additional time costs for encryption and decryption and at the same time does not degrade the system performance of the database. Therefore, the safety analysis and the performance factors which are used are safe and efficient: such as key space, key sensitivity, data security from the attack, calculating speed, information entropy and the correlation coefficients. [3

This paper looks at the method for assessing the security and efficiency of the proposed REA encryption algorithm and is compared with the most common encryption algorithms: DES, 3DES, RC2, AES and Blowfish. Comparison will be shown for these encryption algorithms during the encryption and decryption. The comparison for the safety value will also be shown (used for the measurement if information entropy).

Furthermore, another safety measure is a coefficient of correlation of coded fields with proposed encryption algorithm REA. Results of the experiment show that the encryption and decryption time of proposed encryption algorithm REA has very good performances comparing to the other encryption algorithms. The results from safety measures (information entropy) show that the proposed encryption algorithms REA and AES are safer than DES, 3DES, RC2 and Blowfish.

Compared Algorithm Performance

To give more information about the compared algorithm performances, gathered results from other resources are being considered in this paper. It is concluded that the AES is faster and more efficient than other encryption algorithms.[2] As far as the data transfer is considered , there is a slight difference in the performances of different symmetric key schemes (the most of the resources are spent on the data transfer , not for computing). Even if we have the scenario of data transfer, it would be necessary to use AES scheme if the encrypted data are being saved on the other end and are decrypted several times.

The study shows the safety measure levels for Web programming, considers performance measures of the encryption process in the programme language script with Web browsers. After that, a test simulation follows to get the best encryption algorithm compared to the Web browser. [3] It has been shown that Blowfish and AES have the best performances among the others. Both have better encryption (are stronger against data attack) than the other two. [9]

The study is carried out for various popular algorithms such as DES, 3DES, AES and Blowfish. They are implemented and their effect is compared with the encryption of input data of different content and sizes. Algorithms have been tested on two different hardware platforms to compare their performances.

The testing has been performed on two different machines: PII 266 MHz and P4 2.4 GHz. The results showed that Blowfish had very good performances compared to the other algorithms. Also, it has been showed that AES has better performances than 3DES and DES. We can conclude that 3DES has almost 1/3 throughput from DES-a, or in other words DES needs three times more to process the same amount of data. [5]

Rea Algorithm

New encryption algorithm REA is recommended because of its simplicity, efficiency and safety. It can surpass competitive algorithms. The proposed algorithm REA is a symmetric stream code that can be efficiently used for encryption and data protection. A changeable length key is required, which makes it ideal for data security.

Encryption Algorithm Rea

Algorithm steps for encryption REA are shown through:

- Step 1: Text and key input.
- Step 2: Adding the text key.
- Step 3: Converting previous text into the ASCII code.
- Step 4: Converting previous ASCII code into binary data.
- Step 5: Reverse the previous binary data.
- Step 6: Obtain all 8 bits from previous binary data and obtain ASCII code from it.
- Step 7: Divide previous ASCII code by 4.
- Step 8: Obtain the ASCII code from previous distribution and place it as one sign.
- Step 9: Obtain the rest of the previous distribution and place it as a second sign.
- Step 10: Returning the encrypted data.

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 -A the result(first character) and,
      III. The remainder of the previous A second character.
6. Return (EncryptedData).

Algorithm steps for decryption are shown through:

- Step 1: Enter the text encryption and the key.
- Step 2: Circle the encrypted text to get the ASCII sign code and ad the next sign.
- Step 3: Multiply ASCII code first sign by 4.
- Step 4: Add the next digit (the rest) to the multiplication result operation.
- Step 5: Convert the previous ASCII code into the binary data.
- Step 6: Reverse the previous binary data.
- Step 7: Obtain all 8 bits from previous binary data and obtain ASCII code from it
- Step 8: Convert the previous ASCII code into the text.
- Step 9: Remove the key from the text.
- Step10: Return decrypted data.

```
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 → 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. 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).
```

Safety Factor And Work Analysis

The following factors are used as safe and efficient criteria: key space, key sensitivity, data safety from the attack, computing speed, information and correlation coefficient. [4]

Key Space Analysis

Key space is the total number of different keys that can be used in cryptographic system. Algorithm safety (strength) is the key length function. The longer the key is, the algorithm is more resistant to a successful brutal attack. Key length is universally expressed as the number of bits. [4]

Key length of the N-bit has the key space 2n possibility. From the cryptography point, the size of the key space should not be less than 2100 to ensure a high level of security. [6]

The secret key of the proposed encryption algorithm REA is 256 bit long, can be increased, the key space has about 2256 (1.16x1077) different combinations of secret key. Long key space is sufficient for reliable practical use.

Key sensitivity analysis

Good encryption should be vulnerable to a small change in secret keys. Proposed encryption algorithm REA is vulnerable to a small change in secret keys. If the secret key changes a bit, decrypted data are not performed. [8]

Attack analysis

There are well-known attack methods, as well as brutal force that determines the number of steps and time needed for a successful attack.

I. Attack steps

The attack steps are defined as a number of steps needed for performing the most famous attack. The number of steps can help in deterring the time which could be needed for a successful attack using a specific processor, without the need to actually attack the algorithm. Proposed encryption algorithm REA using the key length 256 bit, can be increased and the attack steps are about 2256 (1.16x1077).

II. Time of the attack

The attack time is defined as the time needed to perform the most famous attack on a certain processor. For an example, machine that works on 2000 (Mops) multiply 60 (seconds/minutes) multiply 24 (hours/day) 365 (days/year) equals 6.3072x1016 operation/year. The attack time in years is performed by dividing the attack steps by the pilot Mtop annually (6.3072x1016 operation in a year). Since the proposed algorithm REA used 256-bit, then the time of attack is approximately 1.839x1060 years.

Speed analysis

This is an important tool for evaluating the efficiency of encryption algorithms that measures the time it takes to encrypt and decrypt the process. Encryption and decoding time of the proposed encryption algorithm REA is shown in experimental results and it is fast enough. [3]

RESULTS

This section examines a typical case study that evaluates the security and efficiency of the proposed encryption algorithm REA and compares it to the most common encryption algorithms: DES, 3DES, RC2, AES and Blowfish. Comparisons were made for these encryption algorithms at computing speed (encryption and decryption time) and secure analysis such as the information entropy and the correlation coefficient. To minimize encryption and decryption time, the cryptosystem should be optimized. Practically, information entropy of the encrypted data is smaller compared to the ideal case. Also, if the correlation coefficient is equal to zero, this means that the encryption text is completely different from the original.

All of the experiments were performed on the laptop IV 2.0 GHz Intel processor with 1 MB cache memory, 1 GB memory and one 120 GB disk. The operating system used was Microsoft Windows 7 Professional. Results were made based on the Microsoft SQL Server 2008 database "Northwind", which contains eight tables. Program Tasks were made by Microsoft Visual C # 2008. In the experiments, using two databases from the database "Northwind", the results are:

a. NorthwindPlaintext does not have encrypted fields, but it is used for encryption and decryption of some fields (Table 1) using the most common algorithms for encryption: DES, 3DES, RC2, AES, Blowfish and proposed algorithm REA.
b. Northwind REA has encrypted fields (Table 1) using proposed encryption algorithm REA.

Table 1. *Name of the encrypted fields*

	Field name	Table name
F1	Contact name	Suppliers
F2	Price per unit	Products
F3	Boat address	Orders
F4	Freight	Orders
F5	Price per unit	Order details
F6	Quantity	Order details
F7	Description	Categories
F8	Notes	Employees
F9	Contact name	Buyers
F10	Contact title	Buyers

The keys used in the encryption data keep confidentiality in the encrypted table with the proposed encryption algorithm REA. This table contains five StrTable fields (encrypted table name), StrField (the name is encrypted field), StrKey (is the encrypted key used in the encryption process, StrAlgo (the name of encryption algorithm) and StrFieldType (type of encrypted field), where the first three fields are encrypted with the proposed REA algorithm. Only the administrator will get these keys using the password.

After the administrator enters the password and selects the required database, he needs to look at the encrypted keys table in the "NorthwindREA" database.

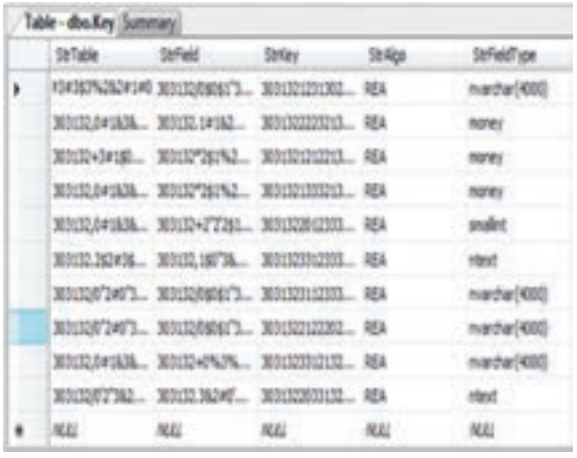


Figure 1. *Table of encrypted keys with proposed REA algorithm in „Northwind_REA" database.*

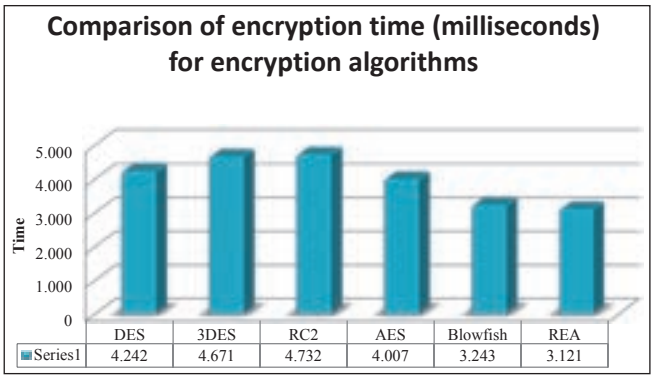
The experiments were encrypted and decrypted in ten different fields (table 2) with the proposed encryption algorithm REA and the calculated execution time for each of them. Then, the average execution time is calculated for each encryption and decryption process.

The comparison results are shown in Table 2 in the encryption time and Table 3 at decoding time.

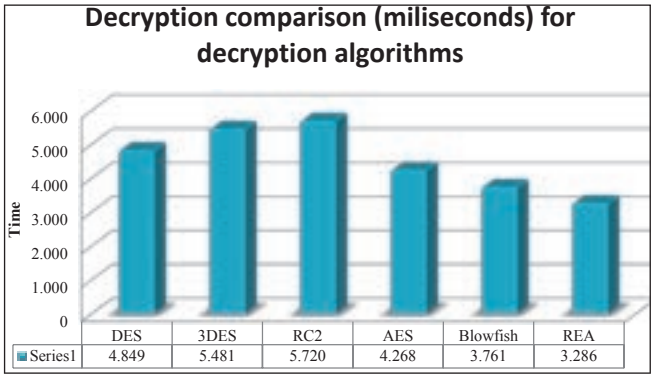
a. First point: The results show the superiority of the REA algorithm compared to other algorithms in terms of encryption and decoding time.
b. Second Point: Blowfish requires less encryption and decoding time than all algorithms except REA.
c. Third point: AES has the advantage compared to the other 3DES, DES i RC2.
d. Fourth point: 3DES has low performances in terms of encryption and decoding time com-

Table 2. *Encryption time comparison (milliseconds) for encryption algorithms*

	DES	3DES	RC2	AES	Blowfish	REA
F1	0.141	0.263	0.342	0.109	0.116	0.104
F2	0.359	0.419	0.395	0.329	0.296	0.266
F3	3.609	4.484	4.594	3.047	2.671	2.521
F4	4.063	4.469	4.513	3.297	2.544	1.718
F5	14.194	14.968	15.234	14.000	11.304	11.297
F6	15.906	17.547	17.328	15.484	12.452	12.360
F7	0.344	0.453	0.449	0.331	0.274	0.265
F8	2.960	3.203	3.531	2.688	2.051	2.005
F9	0.422	0.463	0.487	0.403	0.376	0.335
F10	0.421	0.438	0.442	0.386	0.346	0.334
Average	**4.242**	**4.671**	**4.732**	**4.007**	**3.243**	**3.121**



Graph 1. *Graphical presentation for encryption time comparison (milliseconds) for encryption algorithms*



Graph 2. *Graphic representation of decryption time comparison (milliseconds) for decryption algorithms*

Table 3. *Decryption comparison (milliseconds) for decryption algorithms*

	DES	3DES	RC2	AES	Blowfish	REA
F1	0.125	0.141	0.156	0.135	0.137	0.121
F2	0.343	0.359	0.384	0.344	0.322	0.271
F3	4.672	4.992	5.172	4.212	3.816	3.445
F4	4.313	4.625	4.516	4.103	3.417	1.654
F5	16.687	19.156	20.281	14.266	12.963	12.687
F6	17.797	20.313	21.406	15.125	13.761	11.030
F7	0.359	0.404	0.426	0.359	0.318	0.281
F8	3.312	3.891	3.906	3.319	2.175	2.743
F9	0.443	0.478	0.499	0.421	0.381	0.318
F10	0.438	0.447	0.456	0.398	0.315	0.308
Average	**4.849**	**5.481**	**5.720**	**4.268**	**3.761**	**3.286**

pared to DES. It has also been established that RC2 has low performance in terms of encryption and decryption time compared to other five algorithms.

Overall, the results have shown that the proposed encryption algorithm REA has very good performance compared to other encryption algorithms. It has also been shown that Blowfish and AES have better performance than DES, 3DES and RC2. For the factor information (Security Analysis), secure encryption algorithm should perform a condition about information entropy and this means that the encrypted text should not provide any information about pure text.

Table 4. *Secure (entropy) values comparison for the encryption algorithms.*

	DES	3DES	RC2	AES	BF	REA
F1	5.577	6.678	6.948	7.504	3.861	7.817
F2	4.765	5.872	6.354	7.415	3.801	7.181
F3	5.805	6.878	7.165	7.845	4.405	7.604
F4	4.748	5.847	6.451	7.423	3.791	7.255
F5	4.751	5.846	6.234	7.423	3.783	7.053
F6	4.769	5.841	6.587	7.656	3.799	7.154
F7	5.666	6.754	7.078	7.772	4.212	7.631
F8	5.258	6.339	6.975	7.583	4.163	7.735
F9	5.632	6.702	7.025	7.608	3.998	7.577
F10	5.623	6.667	6.952	7.801	4.358	7.684
Average	**5.259**	**6.342**	**6.777**	**7.603**	**4.017**	**7.469**

The experiments result in safe (entropy) values of encrypted fields with the proposed REA encryption algorithm and comparison with the most common algorithms: DES, 3DES, RC2, AES and Blowfish. The results for this comparison are shown in Table 4, on secure (entropy) values.

Secure (entropy) of encrypted data with the proposed encryption algorithm REA (about 7.469) is less than the ideal case. Therefore, REA design is the data encryption security.

Conclusion

Encryption of sensitive data in databases becomes increasingly important in protection against intruders who bypass the conventional mechanisms of access control and have direct access to the database. The effect and security of the new scheme must be studied systematically. For this purpose it was suggested that these issues be considered in this paper and to contribute to the following:

- We are introducing a new encryption algorithm REA, transferring its advantages and functions compared with other similar encryption algorithms. This limits the costs of added time on encryption and decryption, so the performance of the database system was not reduced.
- Evaluating the efficiency of the proposed REA encryption algorithm and comparing it with the most common encryption algorithms, namely: DES, 3DES, RC2, AES and Blowfish shows the speed of the encryption and decryption process.

The results show the superiority of the REA algorithm compared to other algorithms in terms of encryption and decryption time.

- The safety comparison has been shown (used to measure entropy of information). The results show that safe (entropy) of the encrypted data proposed by the REA encryption algorithm (about 7.469), which is smaller than the ideal case.

Therefore, designing a REA algorithm provides a powerful safety in database encryption.

References

[1] Castano S, Fugini M, Martella G, Samarati P (1995) Database Security, Addison - Wesley, pp. 11.-14.
[2] El – Fishawy N (2007) Quality of Encryption Measurement of Bitmap Images with RC6, MRC6, and Rijndael Block Cipher Algorithms, Proceedings International Journal of Network Security, pp. 5.
[3] Idrus S, Aljunid A (2008) Performance analysis of encryption algorithms text length size on web browsers, International Journal of Computer Science and Network Security, vol.8, pp. 25.
[4] Musheer A, Shamsher M (2009) A New Algorithm of Encryption and Decryption of Images Using Chaotic Mapping , International Journal on Computer Science and Engineering, pp. 46.-48
[5] Salama D, Abdual - Kader H, Hadhoud M (2011) Studying the Effects of Most Common Encryption Algorithms, International Arab Journal of e- Technology, pp. 16.
[6] Weerasinghe T (2012) Secrecy and Performance Analysis of Symmetric Key Encryption Algorithms, International Journal of Information & Network Security, pp. 79.

[7] Zajmović M (2017) Cryptographic data protection methods in databases - example Oracle, Proceedings, University of Novi Sad, Faculty of Applied Management, Economics and Finance Belgrade, Republic of Serbia, pp. 77
[8] Zeghid M, Machhout M, Khriji L, Baganne, Tourki R (2007), A Modified AES Based Algorithm for Image Encryption, World Academy of Science, Engineering and Technology, pp. 206.

Submitted: May 31, 2019
Accepted: June 10, 2019

About the authors

Branko Latinović was born on April 28, 1956 in Prijedor, Bosnia and Herzegovina. He graduated from the Faculty of Economics in Banja Luka in 1980. He completed his master studies at the same faculty in 1994, and successfully defended his doctoral dissertation in 1997. He is the Dean of the College of Information Technologies of Pan-European University "Apeiron" in Banja Luka.

Zoran Ž. Avramović was born in Serbia (Yugoslavia) on September 10th, 1953. He graduated from the Faculty of Electrical Engineering, University of Belgrade. At this Faculty he received a Master's degree, and then a PhD in technical sciences. He is:

- Academician of the Russian Academy of Transport (RTA, St. Petersburg, Russia, since 1995),
- Academician of the Russian Academy of Natural Sciences (RANS, Moscow, Russia, since 2001),
- Academician of the Yugoslav Academy of Engineering (YAE, Belgrade, Serbia, since 2004) (today: Engineering Academy of Serbia, EAS)
- Academician of the Academy of Electrotechnical Sciences of the Russian Federation (AES of the Russian Federation, Moscow, Russia, since 2007)
- Scientific Secretary of the Electrical Engineering Department of the Engineering Academy of Serbia.

Mahir Zajmović was born on July 10, 1986 in Mostar, Bosnia and Herzegovina. He graduated from the Faculty of Information Technologies at the University "Džemal Bijedić" in Mostar in 2010. He successfully completed a joint Master's degree program at the Faculty of Information Technologies of the University "Džemal Bijedić" in Mostar and the Faculty of Economics of the University of Sarajevo in 2014. He is currently a doctoral student.

For citation

Latinović B., Avramović Z.Ž., Zajmović M., Safety Analysis of Reverse Algorithm Encryption in Databases, *JITA – Journal of Information Technology and Applications Banja Luka*, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 9(2019) 1:29-35, (UDC: 004.7.056.55:004.432.2), (DOI: 10.7251/JIT1901029L), Volume 9, Number 1, Banja Luka, june 2019 (1-48), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004