

Reverse Encryption Algorithm - A New Approach For Encryption

Priti V. Bhagat¹, Kaustubh S. Satpute²

¹(Department of Computer Science & Engineering, D.M.I.E.T.R. Sawangi(M), Wardha, Maharashtra, India)

²(Department of Computer Science & Engineering, D.M.I.E.T.R. Sawangi(M), Wardha, Maharashtra, India)

ABSTRACT : In the today's world, security is required to transmit confidential information over the network. Security is also demanding in wide range of applications. Cryptographic algorithms play a vital role in providing the data security against malicious attacks. Encryption of data is an important topic for research, as secure and efficient algorithms are needed that allow optimized encryption and decryption of data. In this paper we propose a new encryption algorithm, called Reverse Encryption Algorithm (REA). Our new encryption algorithm (REA) is simple and fast enough for most applications. REA encryption algorithm provides maximum security and limits the added time cost for encryption and decryption.

Keywords - decryption, encryption, information security, REA

I. Introduction

Cryptography is a science of secret writing. It is the art of protecting the information by transforming it into an unreadable format in which a message can be concealed from the casual reader and only the intended recipient will be able to convert it into original text. Cryptography renders the message unintelligible to outsider by various transformations. Data Cryptography is the scrambling of the content of data like text, image, audio and video to make it unreadable or unintelligible during transmission. Its main goal is to keep the data secure from unauthorized access [22].

Information 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 support is another important way of database security. 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.

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 design the innovative encryption algorithm, called 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. Reverse Encryption Algorithm limits the added time cost for encryption and decryption. We also provide description of the proposed encryption algorithm and its processes.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the proposed encryption algorithm REA. Finally Section 4 presents conclusion

II. 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 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.

III. The Proposed Encryption Algorithm (REA)

We recommend the new encryption algorithm, “Reverse Encryption Algorithm (REA)”, because of its simplicity and efficiency. Reverse Encryption Algorithm limits the added time cost for encryption and decryption. In this section we provide a comprehensive yet concise algorithm.

Our new Reverse Encryption Algorithm is a symmetric stream cipher that can be effectively used for encryption of data. It takes a variable-length key. The REA algorithm encipherment and decipherment consists of the same operations, except the two operations: 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 are presenting the steps of the encryption algorithm of the Reverse Encryption Algorithm REA (Algorithm 1).

The 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.

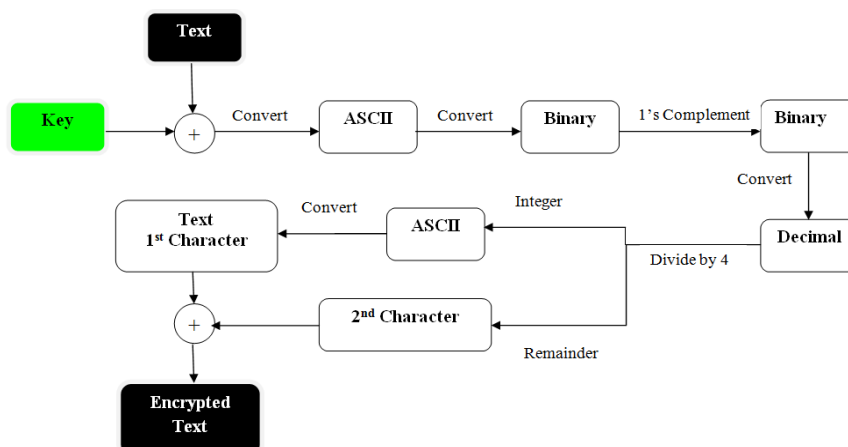
Step4: Convert the previous ASCII code to binary data.

Step5: Find out One’s complement of the previous binary data.

Step6: Gather each 8 bits from the previous binary data and obtain the Decimal value from it. Step7: Divide the previous Decimal value 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.



Algorithm 1: REA_Encryption

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

OUTPUT: Ciphertext (EncryptedData).

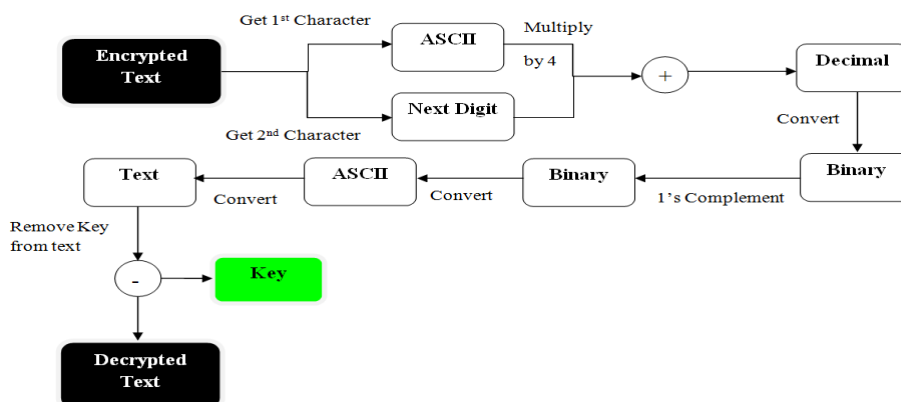
- 1: Add the key to Text (StrKey + StrValue)---> full string (StrFullVlaue).
- 2: Convert the Previous Text(StrFullVlaue) to ASCII code (hexdata).
- 3: For each (byte b in hexdata).
 - a. Convert the Previous ascii code (hexdata) to binary data (StrChar).
 - b. Switch (StrChar.Length).
 Case 7 ---> StrChar = "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: One's complement 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 Decimal Value and,
 - ii. Divide the Decimal Value by 4 → the result(first character) and,
 - iii. 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 Reverse Encryption Algorithm REA (Algorithm 2).

The 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. (Consider result as Decimal value)
- Step5: Convert the previous Decimal value to binary data. Step6:
Find out One's complement of 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.



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. Decimal_value = (EncryptedData[i] * 4) + the next digit(remainder)[i+1].

2: For each (byte b in Decimal_value).

- a. Convert the Previous Decimal_value 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: Find out the One's complement of 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 Example Cipher

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

Text to explain the running methods the proposed algorithm REA.

We have applied our new encryption algorithm REA is on the text, the explanation has been provided below.

The text is: "**Welcome**"

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

key) Encrypted text is:

323130*0&2\$3'0\$0\$2&2

Encipherment:

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

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 Decimal value of it:

206 205 204 168 154. . .

6) Divide the previous Decimal value 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 it as **32**)

205/4 = 51 ---> **3** and the remainder (next digit) = **1** (put it as **31**)

204/4 = 51 ---> **3** and the remainder (next digit) = **0** (put it as **30**)

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

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

7) Encrypted text is (see Figure 3):

"323130*0&2\$3'0\$0\$2&2"

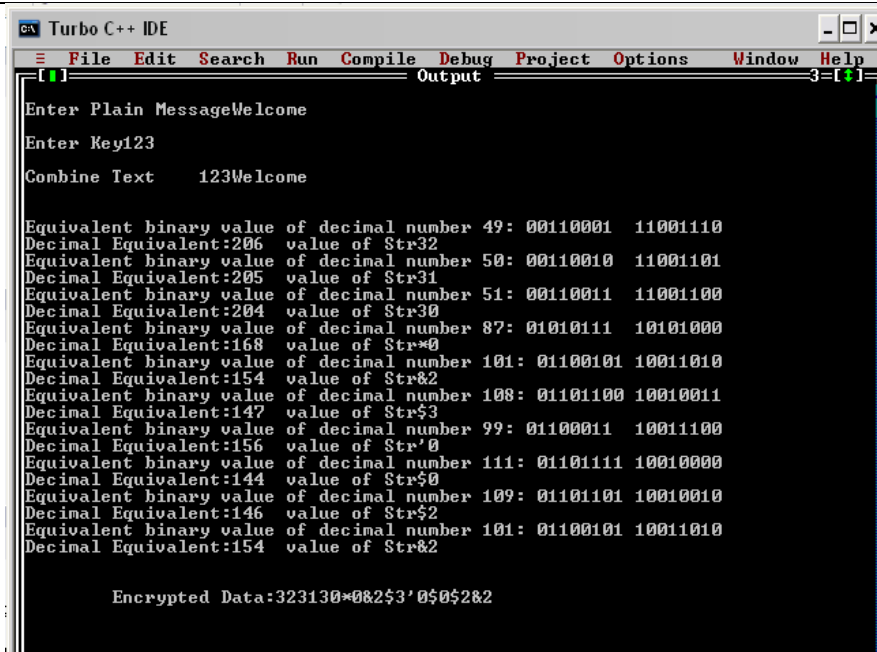
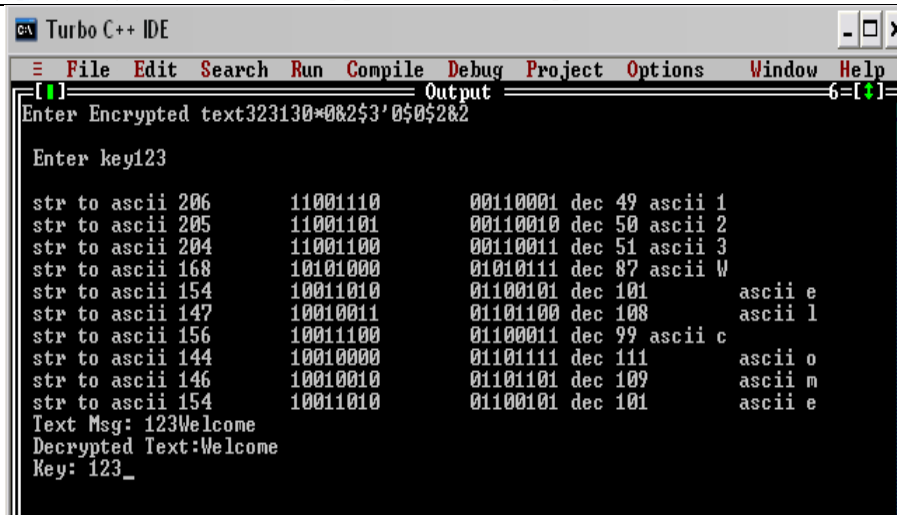


Figure 3: Running the program of the proposed encryption algorithm REA

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 Decimal value is:
 $206 = 51 * 4 + 2$
 The first character = 3 ---> ASCII code is: 51 and the next digit(remainder)= 1 then new Decimal value is:
 $205 = 51 * 4 + 1$
 The first character = 3 ---> ASCII code is: 51 and the next digit(remainder)= 0 then new Decimal value is:
 $204 = 51 * 4 + 0$
 The first character = * ---> ASCII code is: 42 and the next digit(remainder)= 0 then new Decimal value is:
 $168 = 42 * 4 + 0$
 The first character = & ---> ASCII code is: 38 and the next digit(remainder)= 2 then new Decimal value is:
 $154 = 38 * 4 + 2$
- 3) Convert final Decimal value 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: 123>Welcome
- 7) Decrypted text is (see Figure 4):
 "Welcome"



```

Turbo C++ IDE
File Edit Search Run Compile Debug Project Options Window Help
Output
Enter Encrypted text323130*0&2$3'0$0$2&2
Enter key123
str to ascii 206      11001110      00110001 dec 49  ascii 1
str to ascii 205      11001101      00110010 dec 50  ascii 2
str to ascii 204      11001100      00110011 dec 51  ascii 3
str to ascii 168      10101000      01010111 dec 87  ascii W
str to ascii 154      10011010      01100101 dec 101  ascii e
str to ascii 147      10010011      01101100 dec 108  ascii l
str to ascii 156      10011100      01100011 dec 99  ascii c
str to ascii 144      10010000      01101111 dec 111  ascii o
str to ascii 146      10010010      01101101 dec 109  ascii m
str to ascii 154      10011010      01100101 dec 101  ascii e
Text Msg: 123Welcome
Decrypted Text:Welcome
Key: 123_

```

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

IV. CONCLUSIONS

Cryptographic support is an important mechanism of securing important data. In this paper, we introduce a new encryption algorithm, which we call “Reverse Encryption Algorithm (REA)”. Our new encryption algorithm (REA) is simple and fast enough for most applications. We also provide a thorough description of the proposed algorithm and its processes. Our new encryption algorithm REA can reduce the cost time of the encryption/decryption operations and improve the performance.

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 dbms,” *IEEE 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. Doornik 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, “Mv-opes: Multivolume-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*, 4th 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.
- [22] Nidhi Singhal, J.P.S.Raina “Comparative Analysis of AES and RC4 Algorithms for Better Utilization”, *International Journal of Computer Trends and Technology*- July to Aug Issue 2011.