International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 07 | July - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

A New Technique to Generate Ciphertext from Reversed Plaintext Code

Dr. Ummadi. Thirupalu[1], Dr. S. V. Padmavathi Devi[2]

[1, 2]Department of Computer Science and Engineering

[1, 2]Audisankara College of Engineering & Technology, Gudur, Andhra Pradesh, India

Abstract **–** Information security involves safeguarding sensitive data and IT networks from unauthorized access and cyber-attacks. To address this need, various techniques and algorithms have been introduced. In this context, we propose a new technique aimed at protecting information from diverse threats, enhancing user confidentiality, and ensuring high data integrity and availability.

In this paper, we introduce a method for generating cipher text by performing multiple rounds of operations. In the first round, we convert the plaintext into ASCII code, then into hexadecimal, and subsequently reverse the code. These transformed values are then stored in blocks, where various operations are performed to generate the cipher text.

Keywords**:** Cryptography, Security, Encryption, Decryption, Key size.

1. INTRODUCTION

Information security remains a challenging task today. To meet this demand, appropriate ciphers are necessary. Various algorithms have been introduced, depending on parameters such as block size, key size, and techniques like confusion and diffusion. Generally, the block [1] is represented as a block of plain text or cipher text. A key is a piece of information passed to the algorithm to perform various operations on plain text to produce cipher text. When using a symmetric key [2] for encryption, the same key is used to decrypt the cipher text back to the original plain text.

Confusion [3] is the art of designing encryption to hide the relationship between plain text and cipher text, discouraging attackers from locating the key using the cipher text [4]. On the other hand, diffusion [5] makes it difficult for attackers to deduce the plain text from the cipher text.

In this paper, we present both confusion and diffusion techniques applied to a block to produce cipher text. Additionally, we introduce a new cryptosystem that generates a unique cipher for a given plain text, resulting in different cipher texts**.**

2. PROPOSED WORK

Different techniques are used to convert plain text into cipher text through multiple rounds. In this paper, we introduce a method using just two rounds to produce cipher text that is resistant to cryptanalysis.

In the first round, each plain text character is converted into its corresponding decimal ASCII code. This decimal code is then reversed and converted to hexadecimal. In the second round, the hexadecimal code is placed in a suitable square matrix as a block. An XOR operation is performed on the block of hexadecimal codes, using a sequence of numbers as a key to produce the final cipher text. For instance, consider the text "Golden words are not repeated" as the plain text. We then implement the ASCII reverse code process, as shown in the following table.

Table -1: Table which shows plaintext code

Text	Plain text Code	
	Decimal	Hexa-Decimal
G	71	47
o	111	6F
l	108	6C
d	100	64
e	101	65
n	110	6E
	32	20
w	119	77
o	111	6F
r	114	72
d	100	64
s	115	73
	32	20
a	97	61
r	114	72
e	101	65
	32	20
n	110	6E
o	111	6F
t	116	74
	32	20
r	114	72
e	101	65
p	112	70
e	101	65
a	97	61
t	116	74
e	101	65
d	100	64

Table -2: Table which shows reverse plaintext code

Reverse Plain text Code	
Decimal	Hexa-Decimal
17	11
111	6F
801	321
001	1
101	65
011	B
23	17
911	38F
111	6F
411	19B
001	1
511	1FF
23	17
79	4F
411	19B
101	65
23	17
011	B
111	6F
611	263
23	17
411	19B
101	65
211	D3
101	65
79	4F
611	263
101	65
001	1

In the above table, the converted hexadecimal characters are placed into blocks. Key values are then generated as a sequence of natural numbers or a

sequence of even or odd numbers to perform the XOR operation on the block. These numbers are arranged around the block in a clockwise or anti-clockwise direction. This technique is another method to implement confusion operations for cryptanalysis. The XOR operation is performed on the block of code four times using the key values around the block (clockwise or anti-clockwise).

For example, the block code 321 (highlighted in the block) is operated with key values 1, 9, 18, and 22 because these are the key positions around that block code. Similarly, the block code D3 is operated with key values 4, 12, 15, and 19 because these numbers are positioned around that code in the block. This process is repeated for all codes in the block to generate the cipher text.

When using clockwise key positions, the same values are operated with different key position values. For instance, key positions 3, 7, 16, and 24 are used for block code 321, and key positions 6, 10, 13, and 21 are used for block code D3, and vice versa.

Table -3: Clockwise key values

	1	2	3	4	5	6	
24	11	6F	321	1	65	B	7
23	17	38F	6F	19B	1	1FF	8
22	17	4F	19B	65	17	B	9
21	6F	263	17	19B	65	D3	10
20	65	4F	263	65	1		11
19							12
	18	17	16	15	14	13	

Table -4: Anti Clockwise key values

	24	23	22	21	20	19	
1	11	6F	321	1	65	B	18
2	17	38F	6F	19B	1	1FF	17
3	17	4F	19B	65	17	B	16
4	6F	263	17	19B	65	D3	15
5	65	4F	263	65	1		14
6							13
	7	8	9	10	11	12	

The key values for the above blocks range from 1 to 24 natural numbers. Similarly, we can generate key positions using odd or even numbers. These numbers are useful for operating and generating different kinds of cipher text for the specified plain text. The even and odd values for the key positions are shown below. These numbers range from 1 to 47 for odd numbers and 2 to 48 for even numbers. The keys are generated as even or odd for the following blocks. The odd numbers are arranged as anti-clockwise key positions, and the even numbers are arranged as clockwise key positions.

For example, the highlighted code 321 of the block is operated with key values 1, 17, 35, and 43. The highlighted code D3 is operated with key values 7, 23, 29, and 37. Similarly, the code 321 is operated with key values 6, 14, 32, and 48. The code D3 is operated with key values 12, 20, 26, and 42.

By implementing different key positions for a block, we can produce different codes.

Table -5: Clockwise even key values

	2	4	6	8	10	12	
48	11	6F	321	1	65	B	14
46	17	38F	6F	19B	1	1FF	16
44	17	4F	19B	65	17	B	18
42	6F	263	17	19B	65	D3	20
40	65	4F	263	65	1		22
38							24
	36	34	32	30	28	26	

Table -6: Anti Clockwise odd key values

	47	45	43	41	39	37	
1	11	6F	321	1	65	B	35
3	17	38F	6F	19B	1	1FF	33
5	17	4F	19B	65	17	B	31
7	6F	263	17	19B	65	D3	29
9	65	4F	263	65	1		27
11							25
	13	15	17	19	21	23	

Sometimes, all the above key positions are implemented together to generate a different kind of cipher text from the plain text.

3. KEY SIZE

For the proposed work, the key size is defined by the size of the block. Simply put the block size specifies the key size, which is directly related to the block size. The keys are generated as natural numbers, even numbers, or odd numbers, arranged either clockwise or anti-clockwise.

The key values are used to perform the XOR operation on the converted hexadecimal code. Sometimes, multiple key positions are used for the block of data to generate a more robust cipher text than the normally generated cipher text.

4. ENCRYPTION PROCESS

Encryption is the process of converting plain text into cipher text. This paper introduces a new technique for encryption. In this process, plain text is first converted into decimal code, then the decimal code is reversed and converted into hexadecimal. The hexadecimal values are then placed into a block. A key code is generated to perform the XOR operation, producing the final cipher text.

The java code which is used to implement the said task is as follows.

```
import javax.swing.* ;
class RevHex
{
   public static void main(String []args)
   {

          String s;
          intar,i,j,num,n,x;
          introwL[],rowR[],colB[],colT[];
          //variable declaration
          int a[][];

          s=JOptionPane.showInputDialog(null,"Ent
er String to Encrypt");
          char c[]=s.toCharArray();
          n=s.length();
          for(i=0;i<n;i++)                    //process
for block size
              if(i*i>=n)
                  break;
          ar=i;
          a=new int[ar][ar];                          //
memory allocating all arrays
          rowL=new int[ar];
          rowR=new int[ar];
          colB=new int[ar];
          colT=new int[ar];

          x=0;
          br:for(i=0;i<ar;i++)                        //
storing data into block
          {
              for(j=0;j<ar;j++)
              {
          a[i][j]=Integer.parseInt(Integer.toHexStrin
g((int)c[x]),16);
                    x++;
                    if(x>=n)break br;
              }
          }
          num=2;                                  //Key
Position storing
          for(i=0;i<ar;i++){
          rowL[i]=num;num+=2;}
          for(i=0;i<ar;i++){
          colB[i]=num;num+=2;}
          for(i=ar-1;i>=0;i--){
          rowR[i]=num;num+=2;}
          for(i=ar-1;i>=0;i--){
          colT[i]=num;num+=2;}

          int out[][]=new int[ar][ar];
//Performing XOR operation
          for(i=0;i<ar;i++)
          {
               for(j=0;j<ar;j++)

          out[i][j]=(((a[i][j]^rowL[i])^colB[j])^row
R[i])^colT[j];

          }
          s="";
          for(i=0;i<ar;i++)
          //print output cipher text
          {
               for(j=0;j<ar;j++)
                    s+=(char)out[i][j];
          }
          JOptionPane.showMessageDialog(null,s);
   }
}
```

5. DECRYPTION PROCESS

Decryption is the reverse process of encryption, used to retrieve the plain text from the cipher text. To obtain the plain text from the generated cipher text, we perform the same operations in reverse order.

First, perform the XOR operation using the same key positions to retrieve the hexadecimal code. Then convert the hexadecimal code back to decimal code. Reverse the decimal code to obtain the actual plain text..

6. SAMPLE INPUT

The sample input which is used to generate the cipher text by using our newly designed cryptosystem.

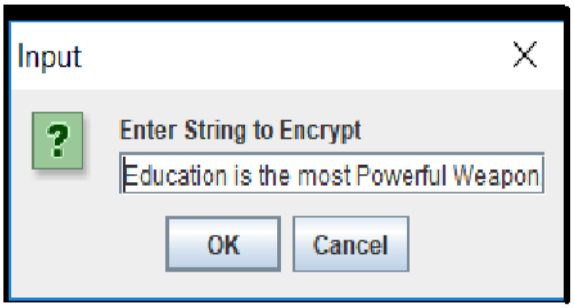

Fig -1: Input box to enter plaintext

7. OUTPUT

The sample output which shows the message which is generated my the algorithm as a ciphertext.

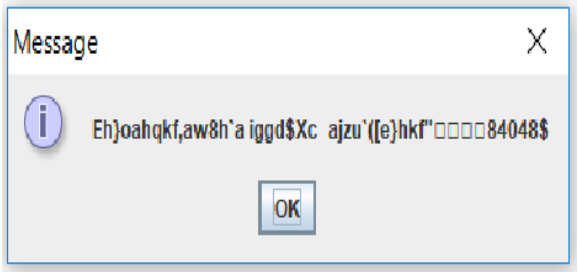

Fig -2: Output which is generated as cipher text.

8. CONCLUSION

This cryptographic technique is closely associated with the symmetric key cryptosystem, where the same key is employed for both encryption and decryption processes. Its design aims to pose challenges for cryptanalysts attempting to break the cipher text. The focus of this paper is to conduct multiple rounds of operations on the block diagonally, utilizing a key, to produce a robust Ciphertext.

References

1. Rebeiro, Chester, Debdeep Mukhopadhyay, and Sarani Bhattacharya. "Modern cryptography." Timing Channels in Cryptography. Springer International Publishing, 2015. 13-35.

2. Qahtan M. Shallal and Mohammad Ubaidullah Bokhari, “A Review on Symmetric Key Encryption Techniques in Cryptography”, International Journal of Computer Applications (0975 – 8887) Volume 147 – No.10, August 2016.

3. Stallings, William. Network security essentials: applications and standards. Pearson Education India, 2007.

4. Schneier, Bruce. Applied cryptography: protocols, algorithms, and source code in C. john wiley & sons, 2007.

5. M.Essaid et. al. “A New Image Encryption Scheme Based on Confusion-Diffusion Using an Enhanced Skew Tent Map”, © 2018 The Authors. Published by Elsevier B.V

6. Raychev, Nikolay. "Classical cryptography in quantum context" Proceedings of the IEEE 10 (2012): 2015.

7. Krasimir Yordzhev, “The Bitwise Operations in Relation to the Concept of Set”, *Asian Journal of Research in Computer Science, 1(4): 1-8, 2018; Article no.AJRCOS.44314.*

8. Kahate, Atul. Cryptography and network security. Tata McGraw-Hill Education, 2013.