A Novel Block Cipher Involving a Key bunch Matrix and a Key-based Permutation and Substitution

Dr. V.U.K.Sastry  
Professor (CSE Dept), Dean (R&D)  
SreeNidhi Institute of Science & Technology, SNIST  
Hyderabad, India

K. Shirishaha  
Computer Science & Engineering  
SreeNidhi Institute of Science & Technology, SNIST  
Hyderabad, India

Abstract— In this paper, we have developed a novel block cipher involving a key bunch matrix supported by a key-based permutation and a key-based substitution. In this analysis, the decryption key bunch matrix is obtained by using the given encryption key bunch matrix and the concept of multiplicative inverse. From the cryptanalysis carried out in this investigation, we have seen that the strength of the cipher is remarkably good and it cannot be broken by any conventional attack.

Keywords- Key bunch matrix; encryption; decryption; permutation; substitution; avalanche effect; cryptanalysis.

I. INTRODUCTION

The development of block ciphers, basing upon a secret key, is a fascinating area of research in cryptography. Though there are several block ciphers, such as Hill Cipher [1], Fiestal Cipher [2], DES [3], together with its variants [4][5], and AES [6]. In all these ciphers, the processes, namely, iteration, permutation and substitution play a vital role in strengthening the cipher. More often, in all these ciphers, the block length and the key length are maintained as 64, 128, 192, or 256 binary bits.

In a recent investigation, we have developed a set of block ciphers [7], [8], [9], “in press” [10], “unpublished” [11], [12], wherein, a secret key bunch matrix plays a prominent role. In all these ciphers, the encryption key bunch matrix contains a set of keys, in which each key is an odd number lying in [1-255]. In all these analyses, the corresponding decryption key bunch matrix, which is also containing odd numbers lying in [1-255], is obtained by using the concept of the multiplicative inverse [4]. In the development of all these block ciphers, the length of the plaintext can be taken as large as possible, at our will, as the size of the key bunch matrix can be chosen as big as possible, in an effective manner. This feature ensures the strength of the cipher in a remarkable way.

In the present investigation, our objective is to develop a novel block cipher, by using the encryption key bunch matrix, and applying a key-based permutation and substitution which strengthen the cipher in a significant manner. The details of the permutation and the substitution processes are presented later.

In what follows, we mention the plan of the paper. In section 2, we discuss the development of the cipher. Further, we present flowcharts and algorithms required in this investigation. Here we deal with the key based permutation and substitution involved in this analysis. In section 3, we offer an illustration of the cipher. In this, we examine the avalanche effect, which acts as a benchmark in respect of the strength of the cipher. In section 4, we make a study of the cryptanalysis. Finally in section 5, we present the computations carried out in this analysis, and arrive at conclusions.

II. DEVELOPMENT OF THE CIPHER

Consider a plaintext P which can be represented in the form of a matrix given by

\[ P = [p_{ij}], i=1 \text{ to } n, j=1 \text{ to } n, \]  

wherein each \( p_{ij} \) is a decimal number lying in [0-255].

Let

\[ E = [e_{ij}], i=1 \text{ to } n, j=1 \text{ to } n, \]  

be the encryption key bunch matrix, in which each \( e_{ij} \) is an odd number lying in [1-255], and

\[ D= [d_{ij}], i=1 \text{ to } n, j=1 \text{ to } n, \]  

be the decryption key bunch matrix, wherein each \( d_{ij} \) is an odd number lying in [1-255]. \( e_{ij} \) and \( d_{ij} \) are connected by the relation

\[ (e_{ij} \times d_{ij}) \mod 256 = 1, \]  

Here it may be noted that the \( d_{ij} \) is obtained corresponding to every given \( e_{ij} \) in an appropriate manner.

The basic equations governing the encryption and the decryption processes of the cipher can be written in the form

\[ C = [c_{ij}] = [e_{ij} \times p_{ij}] \mod 256, i=1 \text{ to } n, j=1 \text{ to } n \]  

and

\[ P = [p_{ij}] = [d_{ij} \times c_{ij}] \mod 256, i=1 \text{ to } n, j=1 \text{ to } n. \]

On assuming that the cipher involves an iteration process, the flowcharts governing the encryption and the decryption can be drawn as shown in Figs. 1 and 2.

In this analysis, \( r \) denotes the number of rounds in the iteration process, and is taken as 16.
The function Substitute(), occurring in the flowchart of the encryption, denotes the key-dependant substitution process, that we are going to describe a little later. The function ISubstitute(), occurring in the decryption process, denotes the reverse process of the Substitute(). The function Mult(), which is in the decryption process, is used to find the decryption key bunch matrix D from the given encryption key bunch matrix E.

The corresponding algorithms for the encryption and the decryption are written as follows.

**Algorithm for Encryption**
1. Read P,E,K,n,r
2. For k = 1 to r do
   
3. For i=1 to n do
   
4. For j=1 to n do
   
5. \( p_{ij} = (e_i \times p_{ij}) \mod 256 \)
   
6. \( P = \{ p_{ij} \} \)
7. \( P=\text{Permute}(P) \)
8. \( P=\text{Substitute}(P) \)
9. \( C=P \)
10. Write(C)

**Algorithm for Decryption**
1. Read C,E,K,n,r
2. D=Mult(E)
3. For k = 1 to r do
4. \( C=\text{ISubstitute}(C) \)
5. \( C=\text{IPermute}(C) \)
6. For i =1 to n do
   
7. For j=1 to n do
   
8. \( c_{ij} = (d_{ij} \times c_{ij}) \mod 256 \)
   
9. \( C=\{ c_{ij} \} \)
10. \( P=C \)
11. Write (P)

We write the elements of this key in a tabular form as shown below.

| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 156| 14 | 33 | 96 |
| 253| 107| 110| 127|
| 164| 10 | 5  | 123|
| 174| 202| 150| 94 |

Here the first row shows the serial number and the second row is concerned to the elements in the key K.

On considering the order of magnitude of the elements in the key, we can write the above table, by including one more row, in the following form

\[
K = \begin{bmatrix}
156 & 14 & 33 & 96 \\
253 & 107 & 110 & 127 \\
164 & 10 & 5 & 123 \\
174 & 202 & 150 & 94 \\
\end{bmatrix}
\]  

(2.7)

We have taken a key K in the form

\[
K = \begin{bmatrix}
117 & 105 & 161 & 1514 & 111 & 43 & 121 & 139 & 76 & 87 & 12 & 3 & 4 & 6 & 164 & 156 & 14 & 33 & 96 \\
253 & 107 & 110 & 127 & 164 & 10 & 5 & 123 & 174 & 202 & 150 & 94 \\
\end{bmatrix}
\]  

Here the 3rd row denotes the order of magnitude of the elements in the key.

The process of permutation, basing upon the key used in this analysis, can be explained as follows. Let \( x_1, x_2, x_3, \ldots, x_{14}, x_{15}, x_{16} \) be a set of numbers. On using the numbers, occurring in the first and third rows of the Table-1, we swap the pairs \( (x_1, x_{12}) \), \( (x_2, x_3) \), \( (x_4, x_6) \), \( (x_5, x_{10}) \), \( (x_7, x_8) \), \( (x_9, x_{13}) \) and \( (x_{14}, x_{15}) \). Here it is to be noted that, \( (x_3, x_4) \) are not swapped, as \( x_3 \) is already swapped with \( x_2 \). Similarly, we do not do any swapping in the case of the numbers \( (x_{15}, x_{11}) \), \( (x_{10}, x_2) \), \( (x_{11}, x_1) \), \( (x_{12}, x_9) \), \( (x_{13}, x_{14}) \), \( (x_{15}, x_{11}) \) and \( (x_{16}, x_5) \). This is the basic idea of the permutation process, which we employ in the case of columns
of numbers as well as rows of numbers occurring in a matrix. For clarity of this process, we refer to the illustration that we are going to do in section 3, a little later.

Let us firstly discuss the process of the key based permutation applied on a plaintext obtained in any round of the iteration process of the encryption. Consider the plaintext \( P = [P_{ij}], i=1 \) to \( n \), \( j=1 \) to \( n \). Let us consider the first two rows of this matrix. On representing each decimal number \( P_{ij} \) in its binary form, and writing the binary bits in a vertical manner, we get a matrix of size 16xn, for these two rows. On assuming that \( n \) is divisible by 16 (for convenience), we can represent these two rows in the form of \( n/16 \) sub-matrices, wherein each one is a square matrix of size 16. Then on swapping the rows (as pointed out in the case of the numbers x1 to x16) and the columns (subsequently one after another), we get the corresponding permuted matrices. After that, by taking the binary bits in a row-wise manner, we convert them into decimal numbers, and write them in a row-wise manner. Thus we get back a matrix of size 2xn. We carry out this process in a similar manner for every pair of rows and have \( n \) columns. Thus we complete the permutation of the entire matrix and get a permuted matrix of size \( nxn \). However if \( n<16 \), the process of swapping is restricted according to the value of \( n \). For example, let us suppose that \( n=4 \). And \( P \) is of the form given by

\[
P = \begin{bmatrix}
198 & 34 & 45 & 12 \\
56 & 92 & 101 & 223 \\
175 & 49 & 245 & 0 \\
211 & 65 & 8 & 100
\end{bmatrix}
\]  

On writing the 16 decimal numbers in terms of binary bits in a column-wise manner, the matrix (2.8) can be represented in the form of a matrix of size 8x16. This is given by

\[
P = \begin{bmatrix}
1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{bmatrix}
\]

(2.9)

Firstly, as suggested by Table-1, we interchange the row pairs (2,3), (4,6), and (7,8). Thus we get

\[
P = \begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

(2.10)

We need not interchange rows any more as we have only 8 rows in this matrix. Now, we interchange the columns following the information in Table-1. This will lead to a matrix of size 8x16, which is given by

\[
P = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\]

(2.11)

This completes the process of the permutation, denoted by the function Permute().

Let us now describe the process of the key-based substitution. We now consider the numbers [0-255] that are occurring in EBCDIC table. These numbers can be represented in the form of a square matrix of size 16 by writing the table in the form

\[
EB(i,j) = [16(i - 1) + j - 1], \ i = 1 \ to \ 16, \ j = 1 \ to \ 16
\]

(2.12)

On using the basic idea of the key-based permutation process, we permute the rows (firstly) and the columns (subsequently), and obtain the substitution matrix, called SB, given by

\[
SB = \begin{bmatrix}
187 & 178 & 181 & 191 & 179 & 183 & 182 & 188 & 185 & 186 & 174 & 184 & 190 & 189 & 188 & 187 \\
183 & 176 & 187 & 181 & 191 & 183 & 182 & 188 & 185 & 186 & 174 & 184 & 190 & 189 & 188 & 187 \\
189 & 193 & 191 & 185 & 186 & 179 & 187 & 185 & 186 & 179 & 187 & 185 & 186 & 179 & 187 & 185 \\
183 & 179 & 187 & 181 & 191 & 183 & 182 & 188 & 185 & 186 & 174 & 184 & 190 & 189 & 188 & 187 \\
189 & 187 & 178 & 181 & 191 & 179 & 183 & 182 & 188 & 185 & 174 & 184 & 190 & 189 & 188 & 187 \\
183 & 178 & 181 & 191 & 179 & 183 & 182 & 188 & 185 & 186 & 174 & 184 & 190 & 189 & 188 & 187 \\
189 & 187 & 178 & 181 & 191 & 179 & 183 & 182 & 188 & 185 & 174 & 184 & 190 & 189 & 188 & 187 \\
183 & 179 & 187 & 181 & 191 & 183 & 182 & 188 & 185 & 186 & 174 & 184 & 190 & 189 & 188 & 187
\end{bmatrix}
\]

(2.13)

The function Substitute() works as follows: On noticing the position of a decimal number (corresponding to a character in the plaintext, at any stage of the iteration process) in the EBCDIC table, we substitute that number in the plaintext by the decimal number occurring in the same position of the substitution matrix.

The functions IPermute() and ISubstitute() denote the reverse processes of the Permute() and the Substitute(), respectively. The function Mult() is used to find the decryption key bunch matrix \( D \) for the given encryption key bunch matrix \( E \).
III. **ILLUSTRATION OF THE CIPHER AND THE AVALANCHE EFFECT**

Consider the plaintext given below.

Dear Brother-in-law! Up to the time that you went abroad, that is a month back, my mother and father promised to give me to you in marriage. They do not want their daughter to go away to this country. They say that they cannot live without my presence along with this in this country. Now they are searching for an Indian match. You are highly qualified. You did your M.Tech. Now you are doing your Doctorate. How can I forget you? I all the while remember your charming personality and your pleasant talk. It is simply impossible for me to forget you and marry someone else. Whatever my father and mother say to me I want to escape from their clutches and reach you as early as possible. I am finishing my final year exams. I have already passed GRE and TOEFL. I would apply for bank loan with the cooperation of your father and get away from this country very soon and join you without any second thought.

Let us focus our attention on the first 16 characters of the plaintext. This is given by

Dear Brother-in-

On using the EBCDIC code, the plaintext (3.2) can be written in the form of a matrix \( P \) given by

\[
P = \begin{bmatrix}
196 & 133 & 129 & 153 \\
64 & 194 & 153 & 150 \\
163 & 136 & 133 & 153 \\
96 & 137 & 149 & 96
\end{bmatrix}.
\]

Let us take the encryption key bunch matrix \( E \) in the form

\[
E = \begin{bmatrix}
21 & 57 & 171 & 39 \\
101 & 67 & 89 & 223 \\
67 & 157 & 171 & 1 \\
37 & 203 & 233 & 17
\end{bmatrix}.
\]

On applying the concept of the multiplicative inverse, we get

\[
D = \begin{bmatrix}
61 & 9 & 3 & 151 \\
109 & 107 & 233 & 31 \\
107 & 181 & 3 & 1 \\
173 & 227 & 89 & 241
\end{bmatrix}.
\]

On using the plaintext \( P \), the encryption key bunch matrix \( E \) and the encryption algorithm, given in section 2, we get the ciphertext \( C \) in the form

\[
C = \begin{bmatrix}
20 & 197 & 152 & 47 \\
247 & 232 & 171 & 142 \\
91 & 154 & 73 & 113 \\
168 & 34 & 170 & 80
\end{bmatrix}.
\]

Now, on using the decryption key bunch matrix \( D \), given by (3.5), the ciphertext \( C \), given by (3.6), and applying the decryption algorithm, we get back the plaintext \( P \), given by (3.3).

Let us now examine the avalanche effect. On replacing the 4th row 2nd column element, 137 by 169, we get a change of one binary bit in the plaintext. On using this modified plaintext, the encryption key bunch matrix \( E \) and applying the encryption algorithm, we get a new ciphertext \( C \) in the form

\[
C = \begin{bmatrix}
176 & 187 & 193 & 16 \\
120 & 5 & 219 & 17 \\
75 & 35 & 72 & 174 \\
252 & 3 & 116 & 221
\end{bmatrix}.
\]

On comparing (3.6) and (3.7), after converting them binary form, we notice that these two ciphertexts differ by 68 bits out of 128 bits. Let us now consider the case of a one bit change in the key bunch matrix \( E \). This can be achieved by replacing 101 (the 2nd row 1st column element of \( E \)) by 116. Now, on using the modified \( E \), the plaintext \( P \), given by (3.3), and applying the encryption algorithm, we get the corresponding ciphertext \( C \) in the form

\[
C = \begin{bmatrix}
204 & 86 & 71 & 1 \\
77 & 69 & 102 & 100 \\
235 & 116 & 221 & 186 \\
45 & 76 & 235 & 186
\end{bmatrix}.
\]

On converting the ciphertexts (3.6) and (3.8) into their binary form, and comparing them, we find that these two ciphertexts differ by 71 bits out of 128 bits.

From the above analysis, we conclude that the cipher is expected to be a strong one.

IV. **CRYPTANALYSIS**

In the literature of the cryptography, the strength of a cipher can be decided by carrying out cryptanalysis. The different attacks that are available for breaking a cipher are

1. Ciphertext only attack (Brute force attack),
2. Known plaintext attack,
3. Chosen plaintext attack, and
4. Chosen ciphertext attack.

Generally every cipher is designed, so that it withstands the first two attacks [4]. However the latter two attacks are examined intuitively and checked up whether the cipher can be broken by those attacks.

Let us now consider the ciphertext only attack. In this cipher, the encryption key bunch matrix is of size \( n^2 \times n \). The key matrix used in the development of the permutation and the substitution is a square matrix of size 4. Hence the size of the key space is

\[
2^{7n^2+128} = (2^{10})^{0.7n^2+12.8} \approx 10^{2.1n^2+38.4}.
\]

If we assume that the time required for the computation of the cipher with one value of the key in the key space is \( 10^{-7} \) seconds, then the time required for the execution of the cipher with all possible values of the key in the key space is
\[ 10^{2.2n^2 + 38.4} \times 10^{-7} = 10^{2n^2 + 31.4} \]
\[ \frac{365 \times 24 \times 60 \times 60}{365 \times 24 \times 60 \times 60} = 3.12 \times 10^{2.2n^2 + 23.4} \text{ years} \]

In this analysis, as we have taken n=4, the time required for the execution assumes the form \(3.12 \times 10^{31.6}\) years. As this is a very large number, it is simply impossible to break this cipher by the brute force attack.

Let us now consider the known plaintext attack. In order to carry out this attack, we know as many pairs of plaintexts and ciphertexts as we require. If we confine our attention to \(r=1\), that is to the first round of the iteration process, then the basic equations governing the cipher are given by

\[ P = [e_{ij} \times p_{ij}] \mod 256, \ i = 1 \text{ to } n, j = 1 \text{ to } n, \]
\[ P = \text{Permute}(P), \]
\[ P = \text{Substitute}(P), \]
\[ C = P \]

As C is known to us, the P on the right side of (4.4) is known. Thus, though P on the left side of (4.3) is known to us, the P on the right side of (4.3) cannot be determined as the Substitute() and the ISubstitute(), which depend upon the key K, are unknown to us. Hence this cipher cannot be broken by the known plaintext attack, even when \(r=1\), as K is not known. However, if an attempt is made to tackle this problem by the brute force attack, that is choosing K in all possible ways, covering the entire key space of the key K, then the time required for developing the functions Permute() and Substitute() can be shown to be

\[ 2^{128} \times 10^{-7} = 3.12 \times 10^{23.4} \text{ years}. \]

As the length of the key K is 128 binary bits. Here, it is assumed that the time required for the computation of Permute() and Substitute() (together with IPermute() and ISubstitute()) takes \(10^{-7}\) seconds. As this time is very large, we firmly conclude that this cipher cannot be broken by the known plaintext attack, even when we supplement it with the brute force attack.

As the equations governing the cipher, are non-linear and highly involved, due to permutation, substitution and modular arithmetic operations, we envisage that it is not possible to choose either a plaintext or a ciphertext for breaking the cipher by the third or the fourth attack.

In the light of the above facts, we conclude that, this cipher is a strong one and it cannot be broken by any conventional attack.

V. COMPUTATIONS AND CONCLUSIONS

In this investigation, we have developed a novel block cipher by using a key bunch matrix. In this, we have made use of a permutation process and a substitution process basing upon a key matrix of size 4x4. The strength of a cipher has increased enormously as we have introduced iteration process and the functions Permute() and Substitute().

The programs required for encryption and decryption are written in Java.

When the size of the plaintext is very large, it is rather tedious to carry out the encryption process by using a key bunch matrix E of size 4x4. Thus, in order to carry out the encryption of the entire plaintext, given in (3.1), we take a key bunch matrix EK of size 16x16. This is taken in the form, given by (5.1).

\[ \begin{bmatrix}
    49 & 163 & 109 & 217 & 133 & 161 & 225 & 89 & 163 & 209 & 225 & 255 & 39 & 31 & 215 & 169 \\
    13 & 227 & 207 & 107 & 207 & 267 & 191 & 161 & 143 & 215 & 29 & 179 & 133 & 45 & 57 & 5 \\
    233 & 211 & 79 & 121 & 91 & 95 & 167 & 89 & 157 & 199 & 111 & 175 & 249 & 71 & 213 & 139 \\
    233 & 195 & 247 & 7 & 211 & 185 & 41 & 243 & 223 & 81 & 83 & 113 & 149 & 27 & 1 & 213 \\
    91 & 129 & 73 & 47 & 187 & 245 & 115 & 143 & 133 & 209 & 31 & 27 & 243 & 39 & 159 & 11 \\
    131 & 185 & 23 & 17 & 187 & 255 & 169 & 97 & 55 & 157 & 149 & 199 & 247 & 85 & 61 & 27 \\
    255 & 209 & 29 & 9 & 41 & 371 & 117 & 145 & 107 & 139 & 91 & 1 & 227 & 87 & 243 & 9 \\
    133 & 93 & 49 & 111 & 115 & 131 & 239 & 63 & 141 & 137 & 193 & 23 & 43 & 193 & 179 & 217 \\
    217 & 97 & 19 & 245 & 113 & 83 & 109 & 159 & 147 & 49 & 225 & 41 & 243 & 103 & 99 & 139 \\
    151 & 143 & 191 & 205 & 91 & 151 & 197 & 137 & 23 & 151 & 105 & 91 & 109 & 91 & 11 & 65 \\
    249 & 39 & 33 & 143 & 69 & 247 & 243 & 53 & 11 & 211 & 99 & 119 & 13 & 19 & 207 & 221 \\
    223 & 101 & 225 & 233 & 61 & 111 & 201 & 149 & 31 & 1 & 55 & 121 & 3 & 175 & 101 & 93 \\
    85 & 61 & 95 & 195 & 33 & 41 & 33 & 71 & 151 & 43 & 93 & 233 & 195 & 159 & 11 & 97 \\
    175 & 93 & 19 & 99 & 59 & 73 & 167 & 127 & 247 & 95 & 135 & 209 & 29 & 55 & 25 & 163 \\
    231 & 215 & 131 & 237 & 131 & 93 & 255 & 188 & 211 & 107 & 77 & 47 & 91 & 249 & 39 & 105 \\
    75 & 225 & 189 & 41 & 75 & 251 & 183 & 79 & 199 & 101 & 95 & 179 & 63 & 189 & 67 & 19 \\
\end{bmatrix} \]

The plaintext given in (3.1) is containing 907 characters. This can be divided into 4 blocks, wherein each block is containing 256 characters. However, we have appended 117 zeroes characters so that we make the last block a complete block. Now, on using K and EK, given in (2.7) and (5.1), and the encryption process, given in section 2, four times, we get the cipher text in the form, given in (5.2).

In order to send the size key bunch matrix EK, in a secret manner, let us encrypt this one by using E as the key bunch matrix. Thus we arrive at the ciphertext corresponding to EK as shown in (5.3).

It is to be noted here, that the sender has to send the ciphertext corresponding to entire plaintext, the number of characters added in the last block, and the ciphertext corresponding to EK to the receiver. Further the sender has to provide E and K in a secret manner.

From the above analysis, we notice that this cipher is a strong one and it can be applied for the transmission of a plaintext of any length in a secured manner. It may also be noted here that this cipher is very much useful in encoding black and white images and color images.

REFERENCES

[1] Lester Hill, (1929), “Cryptography in an algebraic alphabet”, (V.36 (6), pp. 306-312.), American Mathematical Monthly.
[2] Fiestel H., Cryptography and Computer Privacy, Scientific American, May 1973.
[3] National Bureau of Standards NBC FIPS PUB 46 “Data Encryption Standard (DES)”, US Department of Commerce, January 1977.
[4] William Stallings: Cryptography and Network Security: Principle and Practices", Third Edition 2003, Chapter 2, pp. 29.
[5] Tuchman, W., “ Hellman presents no Shortcut Solutions to DES”, IEEE Spectrum, July, 1979.
[6] Daemen J., Rijman V., “Rijndael, The Advanced Encryption Standard (AES)”, Dr. Dobb’s Journal, vol. 26, No. 3, March 2001, pp. 137-139.
[7] Dr. V.U.K. Sastry, K.Shirisha, “A Novel Block Cipher Involving a Key Bunch Matrix”, in International Journal of Computer Applications (0975 – 887) Volume 55– No.16, Oct 2012, Foundation of Computer Science, NewYork, pp. 1-6.

[8] Dr. V.U.K. Sastry, K.Shirisha, “A Block Cipher Involving a Key Bunch Matrix and Including Another Key Matrix Supplemented with Xor Operation”, in International Journal of Computer Applications (0975 – 887) Volume 55– No.16, Oct 2012, Foundation of Computer Science, NewYork, pp.7-10.

Dr. V.U.K. Sastry, K.Shirisha, “A Block Cipher Involving a Key Bunch Matrix and Including another Key Matrix Supplemented With Modular Arithmetic Addition”, in International Journal of Computer Applications (0975 – 887) Volume 55– No.16, Oct 2012, Foundation of Computer Science, NewYork, pp.11-14.

[9] Dr. V.U.K. Sastry, K.Shirisha, “A novel block cipher involving a key bunch matrix and a permutation”, International Journal of Computers and Electronics Research (IJCER), in press.

[10] Dr. V.U.K. Sastry, K.Shirisha, “A block cipher involving a key bunch matrix, and a key matrix supported with xor operation, and supplemented with permutation”, unpublished.

[11] Dr. V.U.K. Sastry, K.Shirisha, “A block cipher involving a key bunch matrix, and a key matrix supported with modular arithmetic addition, and supplemented with permutation”, unpublished.

223 241 161 13 58 52 154 202 32 81 6 150 237 156 161 183 121 39 196 90 88 91 197 252 96 78 118 17 201 95 137 127 189 132 82 3 45 208 66 85 62 158 217 227 42 11 133 104 129 160 72 21 246 93 91 29 75 113 73 79 246 108 54 97 88 219 168 114 10 133 194 178 249 91 152 182 241 251 74 148 233 148 80 51 235 204 235 115 239 223 38 40 24 64 34 65 105 227 176 240 113 3 12 74 151 190 81 165 7 112 111 241 130 153 4 158 188 202 15 197 52 225 121 52 84 3 214 24 198 36 184 60 138 1 46 120 200 16 180 52 117 21 62 168 203 43 90 35 37 198 133 38 136 58 192 176 215 28 171 253 60 173 43 77 169 151 148 188 134 188 76 5 211 62 207 55 165 156 127 144 210 226 82 208 186 55 45 44 114 144 234 20 44 141 63 218 151 48 210 37 50 188 78 100 66 83 120 225 202 89 201 175 183 99 58 125 171 78 232 81 9 110 238 185 21 223 53 6 66 165 35 185 41 42 81 35 66 150 201 104 68 244 63 124 221 208 186 126 236 14 230 11 184 224 209 58 34 190 74 206 29 42 171 196 57 131 13 226 53 29 140 190 16 149 250 131 103 182 200 194 3 183 181 19 62 128 177 61 107 217 242 176 61 164 124 112 177 56 234 167 60 190 102 152 2 205 77 188 160 40 243 72 13 118 184 20 27 28 216 119 150 93 173 227 45 85 4 13 109 83 190 183 254 44 116 147 247 68 119 196 192 125 251 245 202 227 175 255 240 28 233 185 137 237 225 186 187 144 82 220 85 56 15 82 136 86 86 211 200 81 131 34 167 119 252 109 57 28 145 75 189 155 130 226 176 52 184 200 182 153 199 58 219 222 95 55 46 150 123 49 254 250 36 137 218 149 259 150 148 257 225 194 139 153 169 71 12 106 183 133 195 232 237 124 244 121 153 149 15 111 250 35 126 55 101 97 218 15 252 68 43 53 199 156 13 193 191 131 197 69 175 193 105 109 150 48 217 119 165 196 200 93 198 2 80 242 122 48 126 88 249 176 21 96 189 108 223 20 103 0 212 120 170 72 142 205 146 144 218 118 24 199 36 133 143 97 3 1 138 154 44 133 195 9 167 180 153 230 18 232 230 129

Authors Profile

Dr. V. U. K. Sastry is presently working as Professor in the Dept. of Computer Science and Engineering (CSE), Director (SCDSI), Dean (R & D), SreeNidhi Institute of Science and Technology (SNIST), Hyderabad, India. He was Formerly Professor in IIT, Kharagpur, India and worked in IIT, Kharagpur during 1963 – 1998. He guided 14 Ph.Ds, and published more than 86 research papers in various International Journals. He received the Best Engineering College Faculty Award in Computer Science and Engineering for the year 2008 from the Indian Society for Technical Education (AP Chapter), Best Teacher Award by Lions Clubs International, Hyderabad Elite, in 2012, and Cognizant- Sreenidhi Best faculty award for the year 2012. His research interests are Network Security & Cryptography, Image Processing, Data Mining and Genetic Algorithms.

K. Shirisha is currently working as Associate Professor in the Department of Computer Science and Engineering (CSE), SreeNidhi Institute of Science & Technology (SNIST), Hyderabad, India, since February 2007. She is pursuing her Ph.D. Her research interests are Information Security and Data Mining. She published three research papers in International Journals. She stood University topper in the M.Tech.(CSE).

www.ijacsa.thesai.org
