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. Shirisha
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In a recent investigation, we have developed a set of block ciphers [7], [8], [9], “in press” [10], “unpublished” [11], [12], wherein, a secret key bunch matrix plays a prominent role.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

$$P = [\, p_{ij} \,],\ i=1 \text{ to } n,\ j=1 \text{ to } n, \tag{2.1}$$

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

Let

$$E = [\, e_{ij} \,],\ i=1 \text{ to } n,\ j=1 \text{ to } n, \tag{2.2}$$

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

$$D = [\, d_{ij} \,],\ i=1 \text{ to } n,\ j=1 \text{ to } n, \tag{2.3}$$

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

$$(e_{ij} \times d_{ij}) \bmod 256 = 1, \tag{2.4}$$

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

$$C = [\, c_{ij} \,]=[\, e_{ij} \times p_{ij} \,] \bmod 256,\ i=1 \text{ to } n,\ j = 1 \text{ to } n \tag{2.5}$$

and

$$P = [\, p_{ij} \,]=[\, d_{ij} \times c_{ij} \,] \bmod 256,\ i=1 \text{ to } n,\ j = 1 \text{ to } n. \tag{2.6}$$

On assuming that the cipher involoves 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

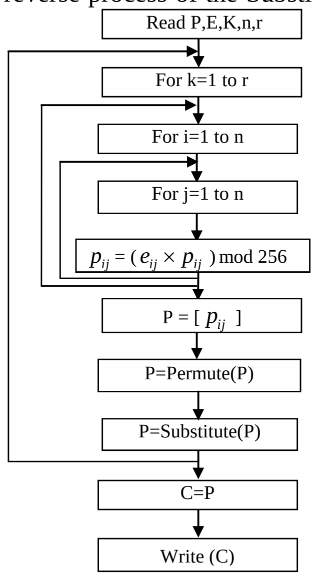

Figure 1. Flowchart for Encryption

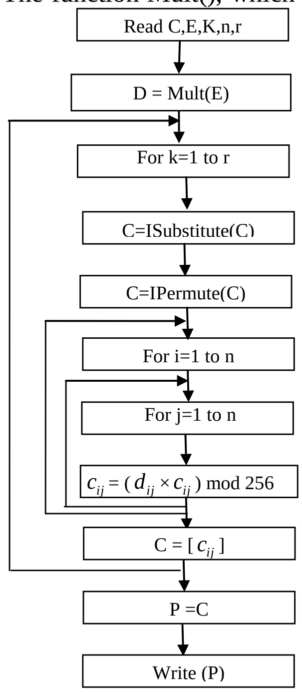

Figure 2. Flowchart for Decryption

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. $p_{ij} = (e_{ij} \times p_{ij}) \bmod 256$
   }
   }
6. P=[p_{ij}]
7. P=Permute(P)
8. P=Substitute(P)
   }
8. C=P
9. 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=ISubstitute(C)
5. C=IPermute(C)
6. For i =1 to n do
   {
7. For j=1 to n do
   {
8. $c_{ij} = (d_{ij} \times c_{ij}) \bmod 256$
   }
   }
9. C=[c_{ij}]
   }
10. P=C
11. Write (P)

To have a clear insight into the key dependent permutation process and key dependent substitution process, which we are adopting in this analysis, let us consider a typical example. Let us take a key K in the form

$$K = \begin{bmatrix} 156 & 14 & 33 & 96 \\ 253 & 107 & 110 & 127 \\ 164 & 10 & 5 & 123 \\ 174 & 202 & 150 & 94 \end{bmatrix} \quad (2.7)$$

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

TABLE I. RELATION BETWEEN SERIAL NUMBERS AND NUMBERS IN ASCENDING ORDER

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
12	3	4	6	16	7	8	10	13	2	1	9	14	15	11	5

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_{16}), (x_7, x_8), (x_9, x_{13}) and (x_{14}, x_{15}). Here it is to be noted that, (x3, x4) are not swapped, as x3 is already swapped with x2. Similarly, we do not do any swapping in the case of the numbers (x_3, x_4), (x_6, x_7), (x_8, x_{10}), (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 2×n.We carry out this process in a similar manner for every pair of rows and having 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

$$P = \begin{bmatrix} 198 & 34 & 45 & 12 \\ 56 & 92 & 101 & 223 \\ 175 & 49 & 245 & 0 \\ 211 & 65 & 8 & 100 \end{bmatrix} \qquad (2.8)$$

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

$$P = \begin{bmatrix} 1&0&0&0&0&0&0&1&1&0&1&0&1&0&0&0 \\ 1&0&0&0&0&1&1&1&0&0&1&0&1&1&0&1 \\ 0&1&1&0&1&0&1&0&1&1&1&0&0&0&0&1 \\ 0&0&0&0&1&1&0&1&0&1&1&0&1&0&0&0 \\ 0&0&1&1&1&1&0&1&1&0&0&0&0&0&1&0 \\ 1&0&1&1&0&1&1&1&1&0&1&0&0&0&0&1 \\ 1&1&0&0&0&0&0&1&1&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&1&1&1&1&1&0&1&1&0&0 \end{bmatrix} \qquad (2.9)$$

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

$$P = \begin{bmatrix} 1&0&0&0&0&0&0&1&1&0&1&0&1&0&0&0 \\ 0&1&1&0&1&0&1&0&1&1&1&0&0&0&0&1 \\ 1&0&0&0&0&1&1&1&0&0&1&0&1&1&0&1 \\ 1&0&1&1&0&1&1&1&1&0&1&0&0&0&0&1 \\ 0&0&1&1&1&1&0&1&1&0&0&0&0&0&1&0 \\ 0&0&0&0&1&1&0&1&0&1&1&0&1&0&0&0 \\ 0&0&1&0&0&0&1&1&1&1&1&0&1&1&0&0 \\ 1&1&0&0&0&0&0&1&1&0&0&0&1&0&0&0 \end{bmatrix} \qquad (2.10)$$

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

$$P = \begin{bmatrix} 0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0 \\ 0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1 \\ 0&0&0&1&1&0&1&1&1&0&1&1&0&0&1&0 \\ 0&1&0&1&1&1&1&1&0&0&1&1&1&0&0&0 \\ 0&1&0&1&0&1&1&0&0&0&0&0&1&1&0&1 \\ 0&0&0&1&0&0&1&0&1&1&1&0&0&0&0&1 \\ 0&1&0&0&0&0&1&1&1&1&1&0&1&0&1&0 \\ 0&0&1&0&0&0&1&0&1&0&0&1&1&0&0&0 \end{bmatrix} \qquad (2.11)$$

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

$$EB(i,j) = [16(i-1) + j - 1],\ i = 1\ to\ 16,\ j = 1\ to\ 16 \qquad (2.12)$$

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

$$SB = \begin{bmatrix} 187&178&177&181&191&179&183&182&188&185&186&176&184&190&189&180 \\ 43&34&33&37&47&35&39&38&44&41&42&32&40&46&45&36 \\ 27&18&17&21&31&19&23&22&28&25&26&16&24&30&29&20 \\ 91&82&81&85&95&83&87&86&92&89&90&80&88&94&93&84 \\ 251&242&241&245&255&243&247&246&252&249&250&240&248&254&253&244 \\ 59&50&49&53&63&51&55&54&60&57&58&48&56&62&61&52 \\ 123&114&113&117&127&115&119&118&124&121&122&112&120&126&125&116 \\ 107&98&97&101&111&99&103&102&108&105&106&96&104&110&109&100 \\ 203&194&193&197&207&195&199&198&204&201&202&192&200&206&205&196 \\ 155&146&145&149&159&147&151&150&156&153&154&144&152&158&157&148 \\ 171&162&161&165&175&163&167&166&172&169&170&160&168&174&173&164 \\ 11&2&1&5&15&3&7&6&12&9&10&0&8&14&13&4 \\ 139&130&129&133&143&131&135&134&140&137&138&128&136&142&141&132 \\ 235&226&225&229&239&227&231&230&236&233&234&224&232&238&237&228 \\ 219&210&209&213&223&211&215&214&220&217&218&208&216&222&221&212 \\ 75&66&65&69&79&67&71&70&76&73&74&64&72&78&77&68 \end{bmatrix} \qquad (2.13)$$

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. (3.1)

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

Dear Brother-in- (3.2)

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

$$P = \begin{bmatrix} 196 & 133 & 129 & 153 \\ 64 & 194 & 153 & 150 \\ 163 & 136 & 133 & 153 \\ 96 & 137 & 149 & 96 \end{bmatrix}. \quad (3.3)$$

Let us take the encryption key bunch matrix E in the form

$$E = \begin{bmatrix} 21 & 57 & 171 & 39 \\ 101 & 67 & 89 & 223 \\ 67 & 157 & 171 & 1 \\ 37 & 203 & 233 & 17 \end{bmatrix}. \quad (3.4)$$

On applying the concept of the multiplicative inverse, we get

$$D = \begin{bmatrix} 61 & 9 & 3 & 151 \\ 109 & 107 & 233 & 31 \\ 107 & 181 & 3 & 1 \\ 173 & 227 & 89 & 241 \end{bmatrix}. \quad (3.5)$$

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

$$C = \begin{bmatrix} 20 & 197 & 152 & 47 \\ 247 & 232 & 171 & 142 \\ 91 & 154 & 73 & 113 \\ 168 & 34 & 170 & 80 \end{bmatrix}. \quad (3.6)$$

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

$$C = \begin{bmatrix} 176 & 187 & 193 & 16 \\ 120 & 5 & 219 & 17 \\ 75 & 35 & 72 & 174 \\ 252 & 3 & 116 & 221 \end{bmatrix}. \quad (3.7)$$

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

$$C = \begin{bmatrix} 204 & 86 & 71 & 1 \\ 77 & 69 & 102 & 100 \\ 235 & 116 & 221 & 186 \\ 45 & 76 & 235 & 186 \end{bmatrix}. \quad (3.8)$$

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 $n \times n$. 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

$$\frac{10^{2.1n^2+38.4}\times 10^{-7}}{365\times 24\times 60\times 60}=\frac{10^{2.1n^2+31.4}}{365\times 24\times 60\times 60}$$

$$=3.12\times 10^{2.1n^2+23.4} \text{ years}$$

In this analysis, as we have taken n=4, the time required for the execution assumes the form $3.12\times 10^{33.6}$ 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 one, 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

$$P = [\, e_{ij} \times p_{ij} \,] \bmod 256,\ i = 1 \text{ to } n,\ j=1 \text{ to } n, \tag{4.1}$$

$$P = \text{Permute}(P), \tag{4.2}$$

$$P = \text{Substitute}(P), \tag{4.3}$$

and

$$C = P \tag{4.4}$$

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

$$\frac{2^{128}\times 10^{-7}}{365\times 24\times 60\times 60}=3.12\times 10^{23.4}\, years.$$

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).

$$EK=\begin{bmatrix}
49 & 163 & 109 & 217 & 133 & 161 & 225 & 89 & 163 & 209 & 225 & 255 & 39 & 31 & 235 & 169\\
13 & 227 & 207 & 107 & 207 & 67 & 191 & 161 & 143 & 215 & 29 & 179 & 133 & 45 & 57 & 5\\
253 & 211 & 79 & 121 & 91 & 95 & 167 & 89 & 157 & 159 & 111 & 175 & 249 & 71 & 213 & 139\\
233 & 195 & 247 & 7 & 231 & 185 & 41 & 243 & 223 & 81 & 83 & 113 & 149 & 27 & 1 & 213\\
91 & 129 & 73 & 47 & 187 & 245 & 115 & 143 & 153 & 209 & 31 & 27 & 243 & 39 & 159 & 11\\
131 & 185 & 23 & 17 & 187 & 255 & 169 & 97 & 55 & 157 & 149 & 199 & 247 & 85 & 61 & 27\\
255 & 209 & 29 & 95 & 77 & 183 & 117 & 145 & 107 & 139 & 91 & 1 & 227 & 87 & 243 & 9\\
133 & 93 & 49 & 111 & 115 & 131 & 239 & 63 & 141 & 137 & 193 & 23 & 45 & 193 & 179 & 217\\
217 & 97 & 19 & 245 & 113 & 83 & 103 & 159 & 147 & 49 & 225 & 41 & 247 & 193 & 99 & 139\\
151 & 143 & 191 & 205 & 91 & 151 & 197 & 137 & 23 & 151 & 103 & 91 & 109 & 91 & 11 & 65\\
249 & 39 & 33 & 143 & 69 & 247 & 243 & 53 & 11 & 211 & 99 & 119 & 13 & 19 & 207 & 221\\
223 & 101 & 225 & 233 & 61 & 111 & 201 & 149 & 3 & 1 & 55 & 121 & 3 & 175 & 101 & 91\\
85 & 61 & 95 & 195 & 33 & 41 & 33 & 71 & 151 & 43 & 93 & 233 & 193 & 159 & 13 & 97\\
175 & 93 & 9 & 99 & 59 & 73 & 167 & 127 & 247 & 95 & 135 & 203 & 29 & 55 & 25 & 163\\
231 & 215 & 131 & 237 & 131 & 93 & 255 & 181 & 211 & 107 & 77 & 47 & 91 & 249 & 39 & 105\\
75 & 225 & 189 & 41 & 75 & 251 & 193 & 79 & 199 & 101 & 95 & 179 & 63 & 189 & 67 & 19
\end{bmatrix} \tag{5.1}$$

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 encrypting 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] Fiestal H., Cryptography and Computer Privacy, Scientific American, May 1973.

[3] National Bureau of Standards NBS 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 – 8887) 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 – 8887) 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 Supported With Modular Arithmetic Addition", in International Journal of Computer Applications (0975 – 8887) 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.

AUTHORS PROFILE

Dr. V. U. K. Sastry is presently working as Professor in the Dept. of Computer Science and Engineering (CSE), Director (SCSI), 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 PhDs, 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).

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	113	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	140	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	92	159	150	148	194	42	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

96	49	188	112	107	141	222	157	170	205	46	109	178	253	165	222
139	181	252	174	248	98	53	127	218	66	139	137	250	100	150	187
108	151	14	72	145	228	52	53	70	105	19	118	36	191	156	146
92	91	46	174	129	134	28	84	214	192	149	81	53	192	186	15
154	238	238	40	35	232	177	185	167	104	28	48	208	240	93	15
22	57	33	35	108	80	156	75	102	41	230	146	7	207	233	195
238	44	12	225	133	232	13	38	73	103	162	224	112	129	227	153
203	197	72	114	207	99	62	144	43	25	9	33	78	111	84	171
163	174	140	226	76	105	49	52	55	55	78	78	120	67	2	121
73	122	80	143	105	146	148	111	136	29	174	98	78	119	51	229
195	191	32	244	64	42	185	129	215	129	33	4	253	106	132	236
150	135	175	43	43	30	79	76	184	216	135	150	255	160	105	253
216	116	114	9	20	109	72	238	216	14	215	228	172	248	98	27
162	203	160	20	89	234	236	104	233	156	240	151	239	148	68	168
8	161	190	31	14	189	213	1	207	246	69	125	94	13	254	154
132	115	175	134	60	136	18	161	2	52	249	201	39	86	62	122

175	213	230	188	248	27	35	68	34	106	240	15	74	205	3	192
110	131	39	230	166	152	240	255	197	110	230	25	33	96	130	43
184	106	138	210	251	94	208	57	174	201	215	106	108	174	243	175
185	50	151	140	253	90	4	216	206	172	143	243	115	120	45	13
251	101	66	108	54	90	42	250	69	147	82	244	7	252	179	53
246	79	17	51	226	3	176	86	114	154	93	127	85	175	139	80
117	210	13	36	64	52	191	216	132	251	226	96	201	235	189	122
144	9	201	125	213	216	83	64	136	217	242	64	255	26	66	141
214	245	158	201	168	139	68	3	221	20	135	142	208	182	145	192
152	34	210	198	251	191	3	146	82	162	51	157	160	224	65	142
10	175	11	7	194	247	249	194	177	63	246	102	49	206	80	30
97	182	174	42	88	184	216	221	242	61	93	2	195	56	88	186
121	190	103	125	218	102	182	84	59	20	67	116	220	245	157	187
197	238	119	91	129	217	7	121	205	189	158	210	44	189	62	69
208	216	180	176	14	27	146	157	214	11	150	20	19	162	208	139
47	248	48	34	135	186	60	178	108	255	230	254	58	65	30	66

(5.2)

$$
\begin{bmatrix}
113 & 73 & 66 & 92 & 33 & 16 & 91 & 0 & 52 & 245 & 249 & 45 & 45 & 131 & 17 & 48 \\
163 & 158 & 75 & 34 & 247 & 172 & 222 & 169 & 121 & 200 & 217 & 190 & 113 & 118 & 23 & 136 \\
98 & 91 & 235 & 68 & 203 & 52 & 99 & 66 & 36 & 60 & 125 & 77 & 109 & 157 & 33 & 14 \\
101 & 252 & 70 & 162 & 63 & 209 & 94 & 80 & 78 & 75 & 208 & 1 & 119 & 112 & 66 & 3 \\
115 & 55 & 85 & 16 & 102 & 144 & 138 & 114 & 254 & 13 & 61 & 230 & 165 & 215 & 168 & 126 \\
149 & 113 & 194 & 100 & 34 & 60 & 85 & 86 & 117 & 204 & 242 & 107 & 29 & 166 & 100 & 208 \\
247 & 69 & 167 & 204 & 194 & 215 & 235 & 46 & 240 & 52 & 46 & 161 & 53 & 216 & 147 & 195 \\
75 & 223 & 70 & 220 & 1 & 123 & 188 & 9 & 122 & 130 & 106 & 217 & 74 & 225 & 145 & 148 \\
188 & 77 & 47 & 145 & 165 & 250 & 126 & 42 & 175 & 39 & 141 & 45 & 186 & 11 & 78 & 122 \\
124 & 108 & 85 & 97 & 134 & 37 & 232 & 80 & 170 & 252 & 236 & 134 & 228 & 6 & 15 & 229 \\
106 & 242 & 28 & 236 & 187 & 64 & 255 & 132 & 233 & 145 & 78 & 54 & 237 & 17 & 214 & 126 \\
105 & 184 & 24 & 1 & 163 & 238 & 34 & 79 & 142 & 213 & 185 & 81 & 233 & 98 & 6 & 91 \\
109 & 12 & 148 & 237 & 225 & 180 & 125 & 20 & 254 & 196 & 192 & 104 & 21 & 54 & 125 & 40 \\
33 & 15 & 59 & 207 & 172 & 241 & 219 & 196 & 156 & 214 & 230 & 250 & 71 & 163 & 9 & 229 \\
3 & 95 & 140 & 134 & 160 & 30 & 140 & 95 & 94 & 174 & 151 & 224 & 47 & 87 & 52 & 233 \\
34 & 38 & 184 & 252 & 222 & 57 & 78 & 47 & 46 & 3 & 30 & 96 & 108 & 156 & 203 & 26
\end{bmatrix} \tag{5.3}
$$