A Block Cipher using Key based Random Permutations and Key based Random Substitutions

K. Anup Kumar† and S. Udaya Kumar†

SreeNidhi Institute of Science and Technology, Hyderabad, India†

Summary
In this paper, we have developed a large block cipher by introducing the basic concepts of permutations and substitutions. The permutations and substitutions are key based. We have taken the key and the plaintext in the form of numbers and characters respectively. Where, each one is converted into its 8 bit binary equivalent based on its ASCII values. In the process of encryption, we have represented the plaintext as a block of 256 bits and developed a block cipher of 256 bits by using the classical feistel network. In each round, we have performed different key based permutations with the help of the equation that we have derived. Similarly, in each round, we have also permuted the key based S-Boxes. The cryptanalysis carried out at end shows that the cipher cannot be broken by any cryptanalytic attack.

Key words:
Block cipher, plaintext, permutation, substitution, XOR, encryption and decryption, SBox.

1. Introduction

In the development of Cryptography, majority of the block ciphers found in literature, are based upon feistel network. The basic elements of this type undergo a series of diffusions and confusions. This is achieved through permutation and substitution of plaintext that is to be encrypted. In the classical feistel network, which involves a round function, wherein the number of rounds is sixteen, provides good strength to ciphers.

In the present paper, our interest is to develop a large block cipher, using 16 rounds classical feistel network, which makes use of key based random substitutions and key based random permutations. In this analysis, we use a key containing 16 numbers, represented as a block of 128 bits. A plain text of 32 characters is represented as a block of 256 bits, finally gives us a block cipher of 256 bits. We developed a linear equation for permutation which depends on the elements of the key. These permutations are different in different rounds. We have also developed the key based substitution boxes and permuted them in each round. So that, a set of intermediate cipher bits will never enter into the same S-box in two consecutive rounds. In the process of encryption- decryption, we have used the same round function 'F' of our classical feistel cipher. In this paper, since the least significant half of the bits of plaintext is exactly equal to the no. of bits of key, We don't need any expansion permutation table separately. We have also discussed the cryptanalysis, which indicates the good strength of the cipher and it is proved by the avalanche affect.

2. Development of Cipher

Consider a key vector 'K' containing 16 numbers. Consider a vector

$D = \{d_0, d_1, \ldots, d_{15}\}$ (2.1)

Where $d_i = K_i \bmod 4$ (2.2)

The elements of vector D allows us to implement various different permutations in different rounds. Let the binary equivalent of these 16 numbers be represented as a matrix k_{16x8}. So that,

$K^0 = \{k_{0x8} + k_{1x8} + \ldots\ldots + k_{14x8} + k_{15x8}\}$ (2.3)

Consider a plaintext vector 'T' containing 32 characters. Let the binary equivalent of these 32 characters be represented as a matrix t_{32x8}. So that,

$T = \{t_{0x8} + t_{1x8} + \ldots\ldots\ldots\ldots + t_{30x8} + t_{31x8}\}$ (2.4)

Here, '+' is the concatenation of bits, t_{ix8} and k_{ix8} are the 8 bits binary equivalent of the i^{th} character of the plaintext vector 'T' and i^{th} number of the key vector 'K' respectively. Let

$C^0 = \{t_{0x8} + t_{1x8} + \ldots\ldots\ldots + t_{30x8} + t_{31x8}\}$ (2.5)

be the initial plaintext. Let $C^1, C^2, \ldots\ldots\ldots C^{15}, C^{16}$ be the 256 bits intermediate cipher text. Such that, C^i is obtained after the i^{th} round during encryption. The linear equation used for Permutation P^i in the i^{th} round is given by

$s = (r + n/2 + d_i) \bmod n$ (2.6)

Such that, 'n' specifies the number of bits on which permutation is applied and d_i is the value which makes permutation distinct in different rounds.

In each permutation P^i,

$r = 0$ to $((n/2) -1)$ (2.7)

We interchange 's^{th}' and 'r^{th}' bits to get required permutation P^i on 'n' bits.

The generation of S-Box from the key is explained in detail with an algorithm in **(** 3.6 **)**. During encryption/decryption, in all the 16 rounds, we have used

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

16 S-Boxes; Each S-Box contains 8 rows and 32 columns; takes 8 bits input and gives 8 bits output. In each round, we have permuted the S-Boxes so that, a set of bits of intermediate cipher will not enter into the same S-Box. So that, there is no scope for cryptanalysis with respect to the substitution boxes.

Let the 256 bits initial plaintext C^0 be divided into two equal parts L^0 and R^0. Such that, L^0 is the most significant 128 bits of C^0 and R^0 is the least significant 128 bits of C^0. Such that Copy the bits of K^1 to kr^1. By using the algorithm given for permutation P^i in (3.3), let us permute the bits of kr^1 by using the value d_0. Thus, round key $kr^1 = P^1 (kr^1, d_0)$.

Next, we permute the S-Boxes according to the algorithm given in (3.4). Now according to the classical feistel network, the first step of the round function is Expansion Permutation table. This is mainly useful if the number of bits in R^0 is less than the number of bits of round key kr^1. Now since R^0 contains 128 bits equal to the number of bits in round key kr^1, we need not use the expansion permutation table and we move to the next step which is XOR of R^0 and round key kr^1. Let $R^1= R^0$ XOR kr^1. Next based on the bits of R^1 we get the 8 bit outputs from the respective S-Boxes. Then permute R^1 and XOR with L^0. Thus, $R^1 = L^0$ XOR (P^1 (R^1)) and $L^1 = R^0$. Now concatenation of all { L^1, R^1 } of respective S-Boxes gives us the intermediate cipher C^1 for the second round. Thus, by using this process for the remaining rounds finally we obtain the cipher C^{16}.

The decryption of cipher is done by using the same round function. We just need to follow the same procedure as explained for encryption. But the key is used in reverse order i.e. kr^{16} to kr^1 and the S-Boxes are reverse permuted in each round as explained in algorithms (3.2) and (3.5).
$C^0 = \{ L^0, R^0 \}$. Now let, K^1 = LeftShift (K^0).

3. Algorithms

3.1. Algorithm for Encryption

BEGIN
1. Read the Key vector K and plaintext vector T
2. Compute the vector D such that $d_i = K_i \bmod 4$
3. Convert the key vector K and plaintext vectors T to binary representation as a matrix k_{16x8} and t_{32x8} respectively.
4. $K^0 = \{ k_{0x8} + k_{1x8} \ldots\ldots\ldots + k_{15x8} \}$
5. $C^0 = \{ t_{0x8} + t_{1x8} \ldots\ldots\ldots + t_{31x8} \}$
6. for i = 0 to 15

{
K^{i+1} = LeftShift (K^i)
Copy K^{i+1} to kr^i
P^{i+1} (kr^{i+1}, d_i) //permute 128 bits; see algo (3.3)
Permute (bx , d_i) // permute S-Boxes; see algo (3.4)
$C^{i+1} = F(kr^{i+1}, C^i, d_i)$
} // end for

END.

3.2. Algorithm for Decryption

BEGIN
1. Read the Cipher C^{16}.
2. Read the last round key K^{16}.
3. Read the vector D = $\{d_0,d_1,d_2\ldots d_{15}\}$.

{
C^i =Switchbox (C^{i+1})
Copy K^{i+1} to kr^i
Permute (kr^i ,d_i)
$C^{i-1} = F(kr^i, C^i, d_i)$
Rightshift (K^{i+1})
C^{i-1}=Switchbox(C^{i-1})
ReversePermute(bx , d_i) // permute S-Boxes;
// see algo (3.5)
} // end for
END

3.3. Algorithm for Permutation

(To permute the key kr^{i+1})

P^{i+1} (kr^{i+1} ,d_i)
BEGIN
1. for r = 0 to (r < n/2)

{
s = (r + n/2 + d_i) mod n
temp = kr^{i+1}_s
$kr^{i+1}_r = kr^{i+1}_s$
kr^{i+1}_s = temp
} // end for
END.

3.4. Algorithm for Permuting S-Boxes

(To permute the S-Boxes during encryption in i^{th} round)

Note: Initial order of 16 S-Boxes before encryption is bx[16] = { 0,1,2,3,………14,15 }.

4. for i = 15 to 0

Permute (bx, d_i)

```
BEGIN
1.  for  r = 0 to ( r  < 8  )
    {
      s =  ( r + 8 + d_i ) mod 16
      temp  =  bx_s
      bx_r =  bx_s
      bx_s =  temp
   } // end for
END.
```

3.5. Algorithm for Reverse Permuting S-Boxes

(To Reverse permute the S-Boxes during decryption in i^{th} round)

Note: Initial order of 16 S-Boxes before decryption will be same as the order used in the last round during encryption.

ReversePermute (bx ,d_i)

```
BEGIN
1.  for  r = 7 to ( r  > = 0  )
    {
      s =  ( r + 8 + d_i ) mod 16
      temp  =  bx_s
      bx_r =  bx_s
      bx_s =  temp
    } // end for
```

3.6. Algorithm to generate Key based Substitution Boxes.

```
BEGIN
1. p = 0

2. for i = 0 to 15
        {
                for j = 0 to 7
                {
                        box p,j = k_i,j

                        box_p+1,j = k_15-i,j
                }
        p = p + 2

        }

3. for  i = 0 to 31
        {
                for j = 5 to 7
                {
                        If  ( box i,j equals to 1 )

                        then

                box i,j =0

                        else

                        box i,j =1
                }
        }

4. for i = 0 to 31
        {
        p = 0

                for j = 0 to 7
                {
                p = p + ( power (2,j) * box_i,7-j )
                }

                Tempbox1_i = p
        }
END.

5.      tempbox2 0,30 = tempbox1_0

        tempbox2 0,31 = tempbox1_1

6.      for j = 0 to 29
        {
                tempbox2 0,j = tempbox1_j
        }

7.      for i = 1 to 15
        {
                tempbox2 i,30 = tempbox2 i-1,0

                tempbox2 i,31 = tempbox2 i-1,1

                for j = 0 to 29
                {
                        tempbox2 i,j = tempbox2 i-1,j
                }


        }

8.      for i = 0 to 15
        {
                for h = 0 to 7
                {
                        for j  to 15
                        {
```

p = (j + 16 + (i mod 2)) mod 32

$tempbox2_{i,j} = tempbox2_{i,j} + tempbox2_{i,p}$

$tempbox2_{i,p} = tempbox2_{i,j} - tempbox2_{i,p}$

$tempbox2_{i,j} = tempbox2_{i,j} - tempbox2_{i,p}$

}

for j = 0 to 31
{
$SBOX^{i}_{h,j}=temp_{i,j}$
}

}

4. Illustration of Cipher

Let the key vector be K = {1, 254, 7, 200, 77, 16, 222, 53, 71, 40, 13, 67, 154, 0, 106, 153 }. (4.1)
Let the vector D = K_i mod 4 = { 1, 2, 3, 0, 1, 0, 2, 1, 3, 0, 1, 3, 2, 0, 2, 1 }. (4.2)
Consider the plaintext vector T = { lets pray together for all of us }. (4.3)
Let the initial order of S-Boxes be denoted by a vector bx[16] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 }. (4.4)
Let the 8 bit binary equivalent of plaintext and key elements be represented by the matrices t_{32x8} and k_{16x8}.
According to (2.3) the key K^0 is as follows.
00000001111111100000011111001000001001101
0001000011011110001101010100011100101000
0000110101000011100110100000000001101010
10011001. (4.5)
According to (2.5) the plaintext C^0 is as follows.
0110110001100101011101000111001100100000
0111000001110010011000010111100100100000
0111010001101111011001110110010101110100
0110100001100101011100100010000001100110
0110111101110010001000000110000101101100
0110110000100000011011110111011000100000
0111010101110011. (4.6)
K^1= kr^1 =LeftShift(K^0).Thus we get K^1 and kr^1 as
0000001111111100000011111001000010011010
0010000110111100011010101000111001010000
00011010100001110011010000000000011010101
00110010. (4.7)

}
kr^1 = P^1 (kr^1, d_0), then we get kr^1 as,
001100011101000010000000111001001111001011 1
01000110011101110100011000000010000001110100
110111011110010001110000011011001101001101 0
10. (4.8)
Let R^0 be the rightmost 128 bits of C^0.
Thus, we get R^0 as follows.
0110010101110010001000000110011001101111 01
1100100010000000110000101101100011011000010
0000011011110110011000100000001110101011100
11. (4.9)
Permute (bx , d_0), and we get the order of S-Boxes as .
bx[16] = { 7, 10, 11, 12, 13, 14, 15, 9, 8, 0, 1, 2, 3, 4, 5, 6 } (4.10)
R^0 = Permute (R^0 ,d_0) // similar to algorithm (3.3)
R^0 = R^0 XOR kr^1.
Now the first eight bits of R^0 goes to $SBox^7$ and the next eight bits of R^0 goes to $SBox^{10}$ and so on according to the order specified in bx[16],i.e. (4.10)
In each SBox, the decimal equivalent of the first 3 bits from the left specifies the row in the SBox, and the decimal equivalent of the remaining 5 bits specifies the column in the SBox.
Let x = $SBox^i$ [row , col], compute the 8 bit binary equivalent of x which is the output of that SBox. Let us concatenate all the outputs of the respective SBoxes and treat it as R^0.
R^0 = permute (R^0 , d_0)
R^1 = L^0 XOR R^0
C^1 = { most significant 128 bits of C^0 , R^1 }. Thus we011001010111001000100000011001100110111 1
0111001000100000011000010110110001101100
0010000001101111011001100010000001110101
0111001111100010011010011111010111000000
0001101101001101010010010101101100011110
1011100101111111101000000110010011100110
0101000101100011. (4.11)
Similarly, by completing the remaining rounds we get the final cipher text C^{16} as follows.

0100100101100010101011001101001100100000
1110110111101111100111100010000000101100
1100011000100111100001111011010010001110
1110011011011000100100100010110000101111
0000000111011011011110101101111011100101
1011011011111000101111001001001100111011
0110100011110000. (4.12)
In the process of decryption we use the same round function but the SBoxes are reverse permuted as explained in algorithm (3.5). Thus we obtain the original plain text from the given cipher text.

5. Cryptanalysis

The various cryptanalytic attacks available in the literature depend upon the facts that, the cipher text is known or pairs of plaintext and cipher text are known or they are chosen in a special manner.

When the cipher text only is known, the breaking of the cipher depends upon the size of the key space, and this is carried out by the brute
get C1 as,
force attack. When the pairs of plaintext and cipher text are known, the cipher can be broken if the key can be determined and this is done by known plain text attack.
Thus, we examine the brute force attack and the known plaintext attack on our cipher to assess the strength of our cipher. Here we show that the brute force attack is formidable and the known plaintext attack leads to a system of equations from which the key cannot be determined due to the unknown term d_i.

Brute Force Attack

As the key matrix is of the order 16 x 8, the size of the key space is

$$2^{128} \approx (2^{10})^{13} \approx (10^3)^{13} \approx (10)^{39} \qquad (5.1)$$

Thus, one cannot break the cipher by applying brute force attack.

Known Plaintext Attack

Let us consider the known plaintext attack. In this case, we have as many plaintext and cipher text
pairs as we require. Through this paper, it is worth noticing the unknown term d_i induced in the equations for permutation and substitution in the repetitive process of feistel cipher.

Firstly, we have used different permutations in different rounds through the equation that we have derived (see (2.6)).

Secondly, the substitution boxes, which used to be static in feistel cipher, are now made random (see 7.3). Now first let us see the known plaintext then we will prove how our algorithm tackles this attack.

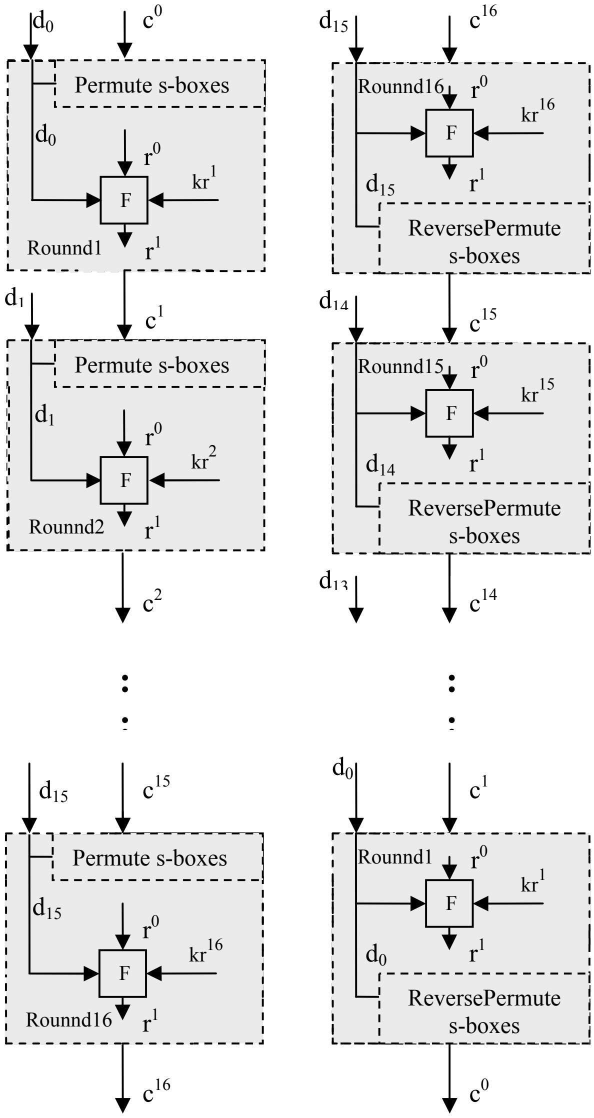
According to classical feistel network, the linear relation between plaintext and cipher text is as follows.

$c^1 \leftarrow F((c^0), kr^1)$ (5.2)
$c^2 \leftarrow F((c^1), kr^2)$ (5.3)
$c^2 \leftarrow F(\ (F(c^0), kr^1), kr^2)$ (5.4)
$c3 \leftarrow F((c^2), kr^3)$ (5.5)
$c3 \leftarrow F((F(\ (F(c^0), kr^1), kr^2), kr^3)$ (5.6)
similarly,
$c^{15} \leftarrow F(c^{14}, kr^{15})$. (5.7)
$c^{16} \leftarrow F(c^{15}, kr^{16})$. (5.8)

Now let us say the know plaintext-ciphertext pair is c^0 and c^{16}. Similarly, we can use as many plaintext-ciphertext pairs as we require. Since c^0 and c^{16} are linearly dependant (see (5.2) to (5.8)) and since the permutations and substitutions are static in all the rounds, The XOR difference of different ciphertext – plaintext pairs will allow us to determine the key.
Now let us see the known plaintext attack on key based permutations and key based substitutions.

$c^1 \leftarrow F((c^0), kr^1), d_0)$ (5.9)
$c^2 \leftarrow F((c^1), kr^2, d_1)$ (5.10)
$c^2 \leftarrow F(\ (F(c^0), kr^1, d_0), kr^2, d_1)$ (5.11)
$c3 \leftarrow F((c^2), kr^3, d_2)$ (5.12)
$c3 \leftarrow F(F(\ (F(c^0), kr^1, d_0), kr^2, d_1), kr^3, d_2)$
(5.13)
similarly, $c^{15} \leftarrow F(c^{14}, kr^{15}, d_{14})$ (5.14)
$c^{16} \leftarrow F(c^{15}, kr^{16}, d_{15})$ (5.15)

attack on classical feistel ciphers and



(a) **(b)**

Encryption using key based permutations & key based substitutions. **Decryption using key based permutations & key based substitutions.**

In this case, even if the plaintext-ciphertext pair c^0 and c^{16} are known, any number of such pairs is of no use mainly because of two reasons. Firstly, the key based permutations are not same in all the rounds. They vary from one round to another round based on permutations, then he should first know the values of d_0, d_1,…,d_{15}. And the values d_0, d_1, ….d_{15} are unknown as long as the key is unknown. Secondly, the order of the key based substitution boxes is not same in all the rounds. They are permuted in each round based on the value of d_i. Therefore, to know which substitution box is used for what bits, one must actually know the value of d_i. And we know d_i is unknown as long as the key secret. Thus the known plaintext attack is also not possible.

Avalanche Effect

Consider the plaintext "lets pray together for all of us" (see (4.3)), we have obtained the cipher text given by (4.12). on changing the first character of the above plaintext from 'l' to 'k' (as we know the ASCII codes of 'l' and 'k' differ in one bit),keeping the key constant, we obtain the corresponding cipher as

0001011101001111101101000100110011111100
0101010101110000010010001001000011101100
1100011000001000100111000111011011111110
0100001110000011100111101111101001000100
1111010010101000100000101111101001011101
1101011011111000101100100001111111100110
0010110110111101. (5.16)

Comparing (4.12) and (5.16), we notice that the two cipher texts differ in 121 bits out of 256 bits. This shows that that the algorithm exhibits strong avalanche effect. the value d_i. And if one has to guess the

Now let us change the key in one bit and keep the plaintext as it is. This is achieved by changing the first number '1' to '0' (since '1' and '0' differ in one bit) in the key vector K given by (4.1). Then we obtain the corresponding cipher as

1001010000100101110010100000111100000000 10
1010100000110111110110010010110011010 1111
1100101000111000001010111001110001001111 0
0001010011011111110111101001000101100111 01
0000101111011101100000101111010111001010 1
0110001111110000101010000100100100100110 0
0010111011. (5.17)

On comparing (4.12) and (5.17),we readily notice that the two ciphers differ in 130 bits out of 256 bits. This shows that, in our encryption algorithm, the permutations and substitution boxes that we have derived from the key exhibit strong avalanche effect.

6. Illustration of permutations

(**To get required key based permutation P^i in the i^{th} round, use**

$d_i = K_i \bmod 4$

$s = (r + n/2 + d_i) \bmod n.$

Let the bits to be permuted are as follows.

0	1	2	3	4	5	6	7
0	1	1	1	1	0	1	0

(6.1)

Since these are the 8 bits.

Here n = 8.

Case 1:

In the i^{th} round, Let $d_i = 0$.

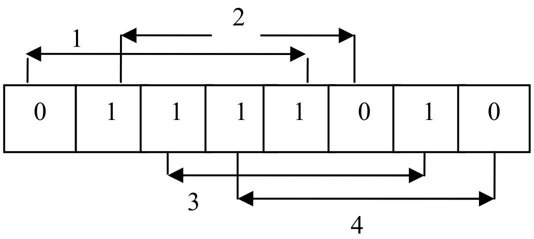
Then according to the equation

$$s = (r + n/2 + d_i) \% n$$

$$r = 0 \text{ to } r < n/2 = \{ 0, 1, 2, 3 \}$$

Then according to (2.6) & (2.7), The corresponding 'S' values will be S = {4, 5 , 6 , 7}

Now interchange the corresponding bits of the 'r^{th}' and 'S^{th}' positions in the following order.



Bits after permutation are

0	1	2	3	4	5	6	7
1	0	1	0	0	1	1	1

Case 2:

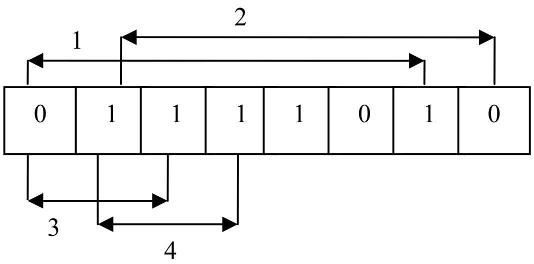
Consider the bits of (6.1)

In the i^{th} round, Let $d_i = 2$.

Thus, n = 8, r = { 0, 1, 2, 3 } and corresponding 's' values will be

$$s = \{ 6, 7, 0, 1 \}$$

Interchange the bits of r^{th} and S^{th} positions in the following order.



Bits after permutation are

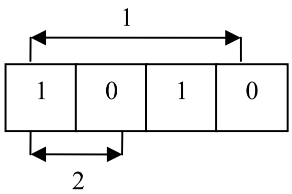
0	1	2	3	4	5	6	7
1	1	1	0	1	0	0	1

Case 3:
Let the number of bits be 4. Thus, n = 4.
Let the bits be as follows.

0	1	2	3
1	0	1	0

(5.2)

In the i^{th} round,
Let $d_i = 1$ and
r = 0 to (r < n/2) = { 0 , 1 }. Then,
according to (2.6) & (2.7), the corresponding 'S' values will be.
S = { 3, 0 }.
Now interchange the corresponding bits of the 'r^{th}' and 'S^{th}' positions in the following order.



Bits after permutation are

0	1	2	3
0	0	1	1

Hence, by following the above procedure, bits of block sizes 64, 128 , 256 etc. or bits of any other higher block sizes can also be permuted. The interesting thing to be observed in this process is the key based value d_i. Since each round will have distinct d_i values, we can generate random permutations in each round without bothering the block size.

7. Generating key based substitution boxes.

Generation of key based substitution boxes:

We need 16 Substitution boxes, each containing 8 rows and 32 columns. Let us expand the elements of key vector K from 16 to 32 numbers. This is done by picking the 16 numbers of K in the reverse order and placing them in between every two consecutive elements of vector K itself. Due to this process, each row of a substitution box will contain a number which is repeated exactly twice. Hence, we get a balanced substitution box.
Let this new vector be K^1 which is as follows.
$K^1 = \{K_0, K_{15}, K_1, K_{14}, K_2, K_{13}, K_3, K_{12}, K_4, K_{11}, K_5, K_{10}, K_6, K_9, K_7, K_8, K_8, K_7, K_9, K_6, K_{10}, K_5, K_{11}, K_4, K_{12}, K_3, K_{13}, K_2, K_{14}, K_1, K_{15}, K_0\}$ (7.1) Now compute the 8 bit binary equivalent of each element in K^1 and then convert the 6^{th}, 7^{th} and 8^{th} bits from 1's to 0's and 0's to 1's respectively.

Next Compute the decimal equivalent of respective 8 bits and place them in a new vector called K^d. Such that
$K^d = \{ K^d_0, K^d_1, \ldots K^d_{31} \}$ (7.2)

Left shift vector K^d twice and place these 32 numbers as the first row in matrix B_{16x32}. Again left shift vector K^d twice and place those 32 numbers as the second row in the matrix B_{16x32}. Continue this process for 14 more times and we get the remaining 14 rows of matrix B_{16x32}.

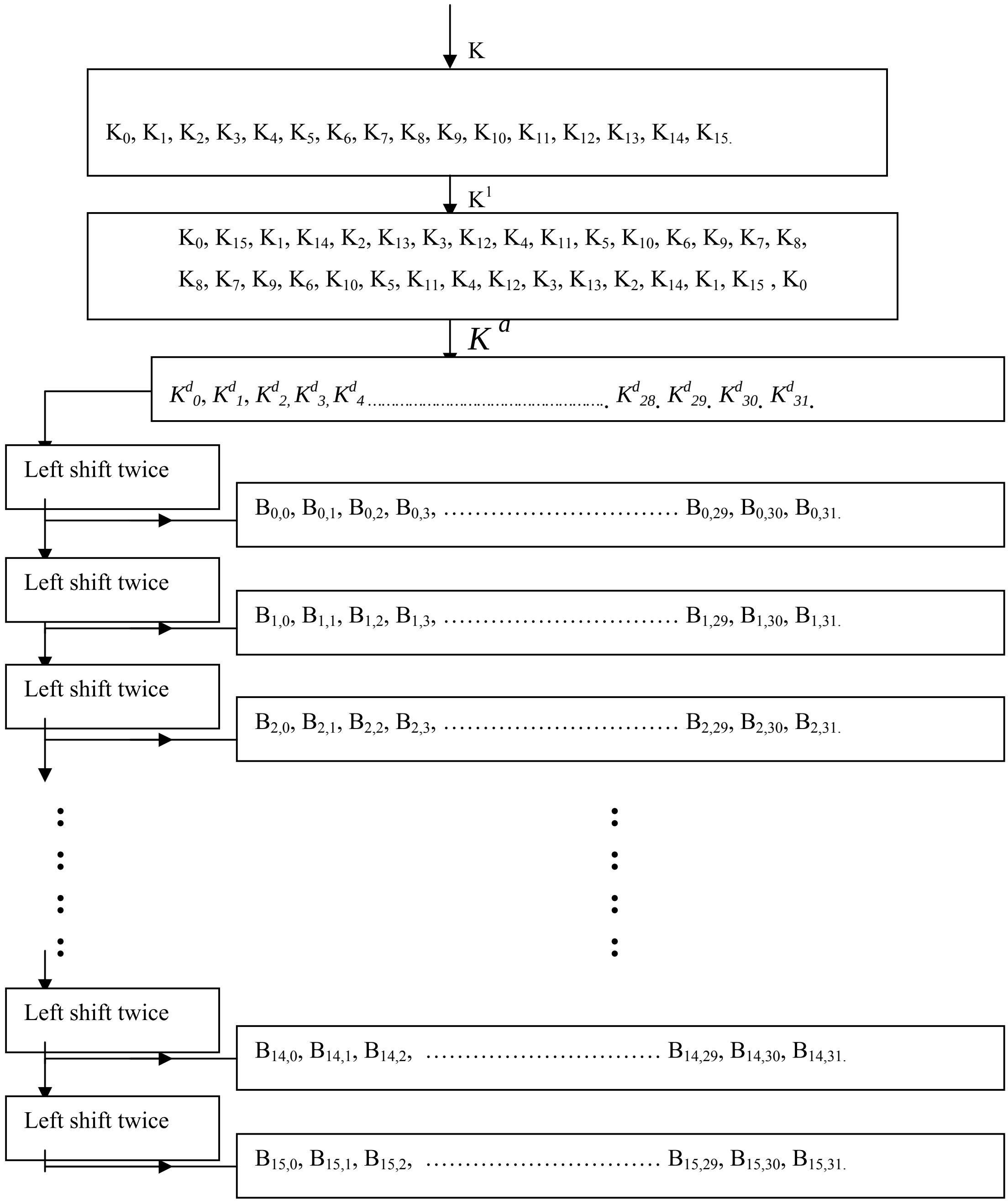
Now we will take the first row of matrix B_{16x32} and permute those 32 numbers by using the following equation.

$$t = (j + n/2 + (i \bmod 2)) \bmod n \qquad (7.3)$$

Here n = 32, since we are permuting 32 numbers. Let j = 0 to (j < n/2). Such that, 't ' and 'j ' indicates the positions at which the elements are to be interchanged.

Let i = 0 to 7. So that, 'i' indicates the row that is to be generated in a substitution box. Such that, For each value of 'i', j = 0 to (j < n/2). Similarly, the other elements of the SBox can be demonstrated.

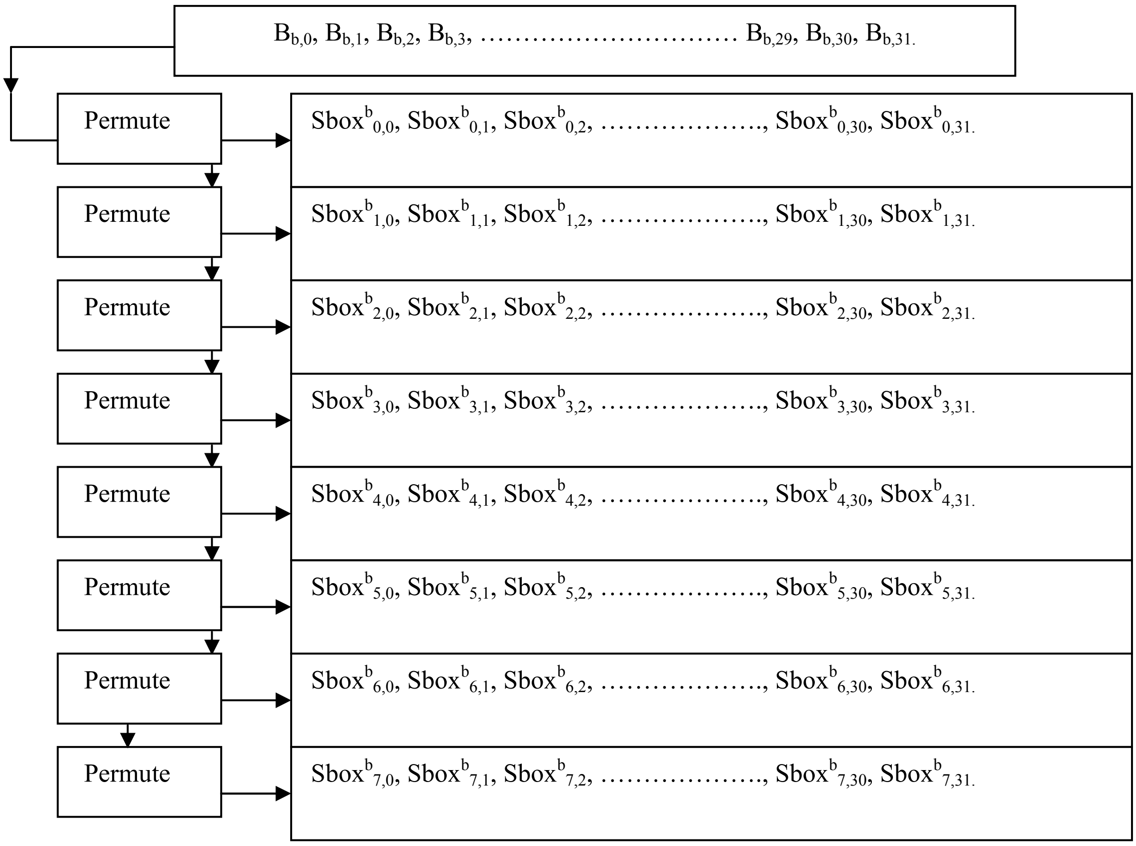
7.1 Illustration of generating key based substitution boxes



7.2 Illustration of generating substitution boxes:

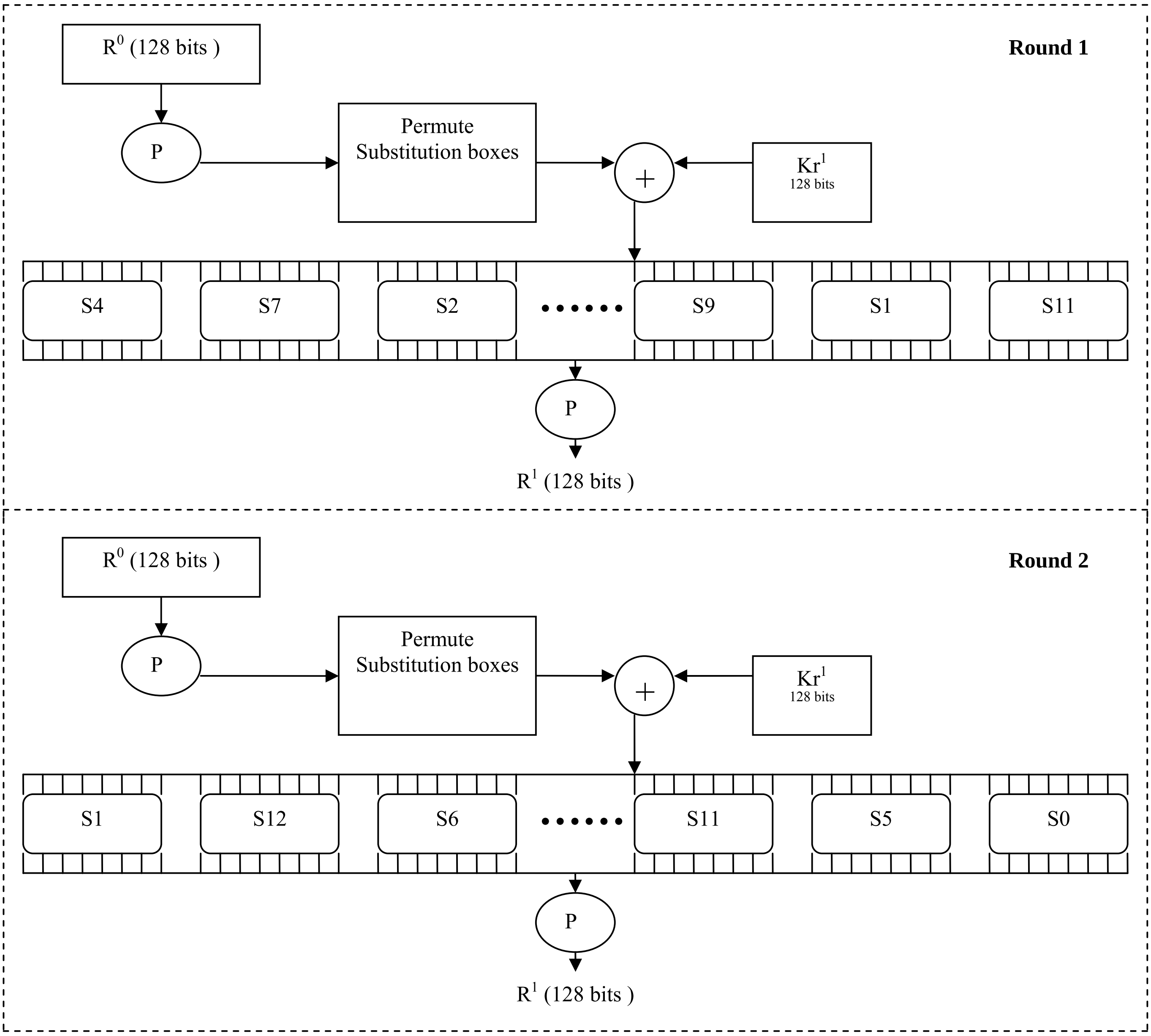
Consider the following code for required permutations in a substitution box.

```
for b = 0  to 15                  // 'b' indicates the substitution box
{
          for i = 0 to 7          //  'i' indicates the row in b^th substitution box
          {
            for j = 0 to 15
             {
                t = ( j  + n/2 + ( i mod 2 ) ) mod n       //Permute // 't' and 'j' indicates the interchange ( B_{i,j} , B_{i,t} )
             positions at which elements are to be interchanged.
             }
               for r = 0 to 31
               {
                         Sbox^b_{i,r} =    B_{i,r}
               }
          }
}
```



7.3 Illustration of permuting substitution boxes:

Let the initial order of substitution boxes be bx[16] = { 0, 1, 2, 3,…14, 15}
After permuting the substitution boxes in the first round,
Let bx = { 4, 7, 2, 0,……..15, 9, 1, 11 }. And
After permuting the substitution boxes in the second round,
Let bx = { 1, 12, 6, 15,……..8, 11, 5, 0 }.



Permutation of Substitution boxes during encryption.
Similarly, during decryption, we perform reverse permutation of substitution boxes.

8.Computational experiments and Conclusions

In this paper, we have developed a block cipher for a block of size 256 bits. The key contains 32 numbers and it is represented as a block of 256 bits by using a matrix. The plaintext which is of 32 characters is also represented as a matrix of 256 binary bits. The development of cipher is essentially represented by feistel network and we have used 16 rounds for encryption. The algorithms given for the encryption- decryption, permutation and substitution are all written in C language.

From the cryptanalysis presented, we have found that the cipher cannot be broken by the brute force attack or known plaintext attack. Moreover since we have used the random substitution boxes and since the permutations and substitutions are based on key, the cryptanalysis is very difficult. Keeping all the above aspects in view, we conclude that the cipher is a very interesting one and it cannot be broken by any cryptanalytic attack.

Acknowledgment:

The authors are very thankful to Prof. Depanwita Roy chaudhury, IIT Kharagpur, for giving her valuable inputs while writing this paper. We are also thankful to the management of SreeNidhi Institute of Science and Technology, for their support and encouragement given during this research work.

References

[1] William Stallings, “Cryptography and Network Security: Principles & Practices”, Third edition, 2003,Chapter 2.

[2] Feistel, H. “Cryptography and Computer Privacy”, Scientific American, vol.228, No.5,pp15-23, 1973.

[3] Feistel, H. , Notz. W. , and Smith. J. “Some Cryptographic Techniques for machine to machine Data Communications”, Proceedings of the IEEE, vol. 63, No. 11, pp, 1545- 1554, Nov. 1975.

[4] “Avalanche Characteristics of Substitution – Permutation Encryption Networks” Tavares S. Heys H. IEEE Transactions on computers 44(9): 1131-1139, 1995.

[5] Shakir M. Hussain and Naim M. Ajlouni, “key based random permutation”, “journal of computer science 2(5): 419-421, 2006. ISSN 1549-3636.