Journal of Computer Science 6 (2): 133-140, 2010
ISSN 1549-3636
© 2010 Science Publications

Block Cipher Involving Key Based Random Interlacing and Key Based Random Decomposition

K. Anup Kumar and V.U.K. Sastry
Department of Computer Science and Engineering,
Sreenidhi Institute of Science and Technology, Ghatkesar,
Hyderabad, 501301, Andhra Pradesh, India

Abstract: Problem statement: The strength of the block ciphers depend on the degree of confusion and diffusion induced in the cipher. Most of the transformations used for this purpose are well known to every one and can be broken by a crypt analyzer. Therefore, in order to counter attack the crypt analyzer, there is a need for better transformations in addition to the existing one. **Approach:** We tried to use key based random interlacing and key based random decomposition for this purpose. So that, a crypt analyzer cannot understand how interlacing and decomposition is done in every round unless the key is known. **Results:** The strength of the cipher is assessed by avalanche effect which is proved to be satisfactory. **Conclusion/Recommendations:** Key based random interlacing and decomposition can be used for introducing confusion and diffusion in block ciphers. The cryptanalysis carried out in this regard shows that the cipher cannot be broken by any cryptanalytic attack.

Keywords: Encryption, decryption, key, random interlacing, random decomposition

INTRODUCTION

In the survey of literature of cryptography, majority of block ciphers are based on the feistel cipher (Tavares and Heys, 1995; Stallings, 2003). In this process, bits of plaintext undergo a series of permutations, substitutions and exclusive OR operations. This creates confusion and diffusion in cipher which is achieved by the classical round function F of feistel structure.

In our recent investigations (Kumar and Kumar, 2008; Kumar and Sastry, 2009); we have demonstrated how a large block cipher of 256 bits can be generated using key based random permutations and involving interlacing and decomposition in feistel structure; providing good strength to cipher.

In the present study, our interest is to develop a block cipher of 256 bits by using a stronger version of interlacing and decomposition. This is accomplished by using key based random interlacing and key based random decomposition. This ensures that interlacing and decomposition creates more confusion as they are different in each round and depends on key. An attacker cannot understand how interlacing and decomposition is done in each round unless the key is known.

MATERIALS AND METHODS

Key based random decomposition used during encryption: Let us consider a block of plaintext 'P' of 256 bits. Let $C^0 = P$ is the initial plaintext. Let B^0_0, B^0_1, B^0_2 and B^0_3 be the 64 bits blocks obtained after decomposition. Let 'K' be the key containing 16 integers.

Let $d_i = K_i \bmod 4$. Such that $d_i = \{0, 1, 2, 3\}$ represents the starting block B^0d_i in ith round into which bits are to be decomposed first. Initially, we place the first 64 bits of 'C^0' in d_i^{th} block by placing the bits from left to right order if K_i is even and by following right to left order if K_i is odd. We then place the next 3 sets of 64 bits of 'C^0' in the respective blocks whose block No $= (d_i + x) \bmod 4$.

Such that, $x = \{1, 2, 3\}$ and by reversing the previous order of placing the bits for every new block that we are decomposing into. Due to key based random decomposition, an attacker cannot guess the order of bits getting into each block unless the key is known. See algorithm

Interlacing during encryption: Let C^i_0, C^i_1, C^i_2 and C^i_3 be the four 64 bit blocks representing the intermediate cipher obtained after i^{th} round encryption.

Corresponding Author: K. Anup Kumar, Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, 501301, Andhra Pradesh, India

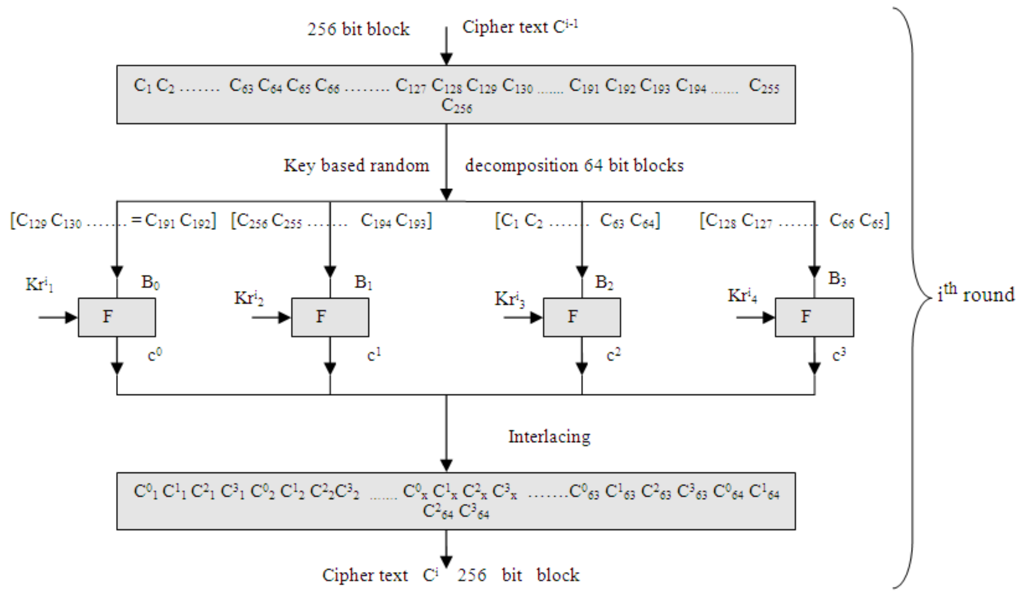
256 bit block
Cipher text C^{i-1}
C1 C2 .... C63 C64 C65 C66 .... C127 C128 C129 C130 ...... C191 C192 C193 C194 ...... C255 C256
Key based random
decomposition 64 bit blocks
[C129 C130 .... = C191 C192]
[C256 C255 .... C194 C193]
[C1 C2 ...... C63 C64]
[C128 C127 .... C66 C65]
Kri1
Kri2
Kri3
Kri4
B0
B1
B2
B3
F
c0
c1
c2
c3
Interlacing
C01 C11 C21 C31 C02 C12 C22C32 ...... C0x C1x C2x C3x C063 C163 C263 C363 C064 C164 C264 C364
Cipher text Ci 256 bit block
ith round

Fig. 1: Illustration of key based random decomposition and interlacing during encryption

We need to combine these four blocks and bring the 256 bits required for next round, which is represented as ‘C^i’. This is done by interlacing which is similar to the one we already published in our previous study (Kumar and Sastry, 2009).

Here, we collect first bit of all four blocks and place them in Ci. Similarly, we collect the second bit from all four blocks and place them in C^i. Continue this process, till all the bits are taken from C^i_0, C^i_1, C^i_2 and c^i_3 and we get the 256 bit C^i for next round, see algorithm.

Consider i^{th} round in the Fig. 1; Let K_i be some even number. Therefore the starting order of placing bits is left to right for block B_{di}. Such that $d_i = (K_i \bmod 4) = 2$.Therefore, first decompose 64 bits into B_2 with Left to right order next into B_3 with right to left order next into B_0 with left to right order and finally into B_1 with right to left order.

Decomposition used during decryption: Decomposition during decryption is simple and it’s the reverse transformation of interlacing used during encryption. Let C^i be the cipher text of 256 bits obtained after i^{th} round. Collect the first four bits from C^i and place one bit each in B^0_0, B^0_1, B^0_2 and B^0_3 respectively. Continue this process till all these bits are taken from C^i and we get the 64 bit blocks B^0_0, B^0_1, B^0_2 and B^0_3 for decryption during i^{th} round, see algorithm.

Key based random interlacing used during decryption: Key based random interlacing used during decryption is the reverse transformation of key based random decomposition done during encryption. Let C^i_0, C^i_1, C^i_2 and C^i_3 be the four 64 bit blocks representing the intermediate cipher obtained after decryption in i^{th} round. We need to interlace the bits of these four blocks to form a 256 bits block for decryption in next round. If c^i_j is the j^{th} block obtained from ith round, then select the block such that, $j = K_i \bmod 4$, collect the 64 bits following left to right order if K_i is even and right to left order if K_i is odd. Place these bits in C^i sequentially.

Select the next three blocks such that:

$$j = ((K_i \bmod 4) + x)$$

where, x = {1, 2, 3} and by reversing the previous order of collecting the bits from each block. Place them sequentially in C^i and we get the 256 bit block to be decrypted in next round. Hence, we get the required plaintext after 16 rounds, see algorithm.

Figure 1 shows the process involved in encryption-decryption in one single round. Similar process is carried out in 16 rounds during encryption-decryption. Due to the key based random decomposition and key based random interlacing demonstrated in Fig. 1 and 2. An attacker cannot trace the way bits are mixed in each round. This can be done only if the entire key sequence K is known.

Development of cipher: Let us consider a block of plaintext ‘P’ consisting of 32 characters. By using the EBCDI code, each character can be represented in terms of 8 bits. Then the 32 characters of plaintext will yield a block of 256 bits represented as C^0.

Let ‘K’ be the key containing 16 integers. Then the 8 bit binary equivalent of these integers will give us a block of 128 bits represented as ‘k’.

Let the first 32 bits of ‘k’ be treated as 'k_1.

The next 32 bits of ‘k’ be treated as k_2.

Similarly, we get two more keys ‘k_3’ and ‘k_4’.

As we use four different blocks B_0, B_1, B_2, B_3 of 64 bit each for encryption, k_1, k_2, k_3, k_4 are used as the keys for these blocks respectively.

We perform the required transformations on k_1, k_2, k_3, k4 to generate the keys for respective rounds denoted as $kr^m{}_1$, $kr^m{}_2$, $kr^m{}_3$, $kr^m{}_4$. Such that if $kr^m{}_i$ is the round key, then ‘i’ indicates the block and ‘m’ indicates the round. Kumar and Kumar, (2008) for required transformations on key.

Decompose C^0 into four blocks of 64 bits each. Let the blocks obtained after key based random decomposition be represented as $B^0{}_0$, $B^0{}_1$, $B^0{}_2$ and $B^0{}_3$.Therefore:

$$B^m{}_i = \leftarrow C^m \rightarrow$$

Where:

m	= The round after which key based random decomposition is performed
i	= The block number; i = 0 to 3
$\leftarrow C^m \rightarrow$	= Key based random decomposition

Encryption in the first round is done in the following way:

$$c^n{}_i = F_{kr^n{}_{i+1}} (B^m{}_i)$$

i = 0 to 3	= i^{th} block
F	= Encryption
$kr^n{}_{i+1}$	= The round key for ‘n^{th}’ round on i^{th} block and n = m+1

After encryption in n^{th} round, we get ciphertext as four blocks $c^n{}_0$, $c^n{}_1$, $c^n{}_2$, $c^n{}_3$.

Next we perform the interlacing after encryption:

$$C^n = > c^n{}_i <$$

Here:

i = 0 to 3	= The cipher block
n = 1 to 16	= The round after which interlacing is performed
$> c^n{}_i <$	= Represents interlacing.

Similarly, if C^{16} is the cipher text obtained after encryption. We continue the process of decryption illustrated in Fig. 4 for sixteen rounds to get the original plaintext.

Algorithms:

Algorithm for encryption: Let K be an array containing 16 integers.

Let d_i be an array containing 16 numbers. Such that, $d_i = K_i \bmod 4$ such that, $d_i = \{0, 1, 2, 3\}$.

```
BEGIN
C^0 = P // initialize 256 bits plaintext
for i = 1 to 16
{
  for j = 1 to 4
  {
     $B^{i-1}{}_{j-1} = \leftarrow C^{i-1} \rightarrow$ // Key based random
                                 Decomposition
  }
  for j = 0 to 3
  {
     $C^i{}_j = F_{kr^i{}_{j+1}} ( B^{i-1}{}_j )$ // Encryption
  }
  for j = 0 to 3
   {
      $C^i = > C^i{}_j <$ // Interlace
   }
}

END
```

Algorithm for decryption:

```
BEGIN

$C^{16}$ = cipher text // initialize 256 bits cipher text
for i = 16 to 1
{
   for j = 0 to 3
 {
 $B^i{}_j = < C^i >$ // Decompose
 }
   for j = 0 to 3
{
 $C^{i-1}{}_j = F_{kr^i{}_{j+1}} ( B^i{}_j )$ // Encryption
}
   for j = 0 to 3
 {
 $C^{i-1} = \rightarrow C^{i-1}{}_j \leftarrow$ // Key based random Interlacing.
 }
}

END
```

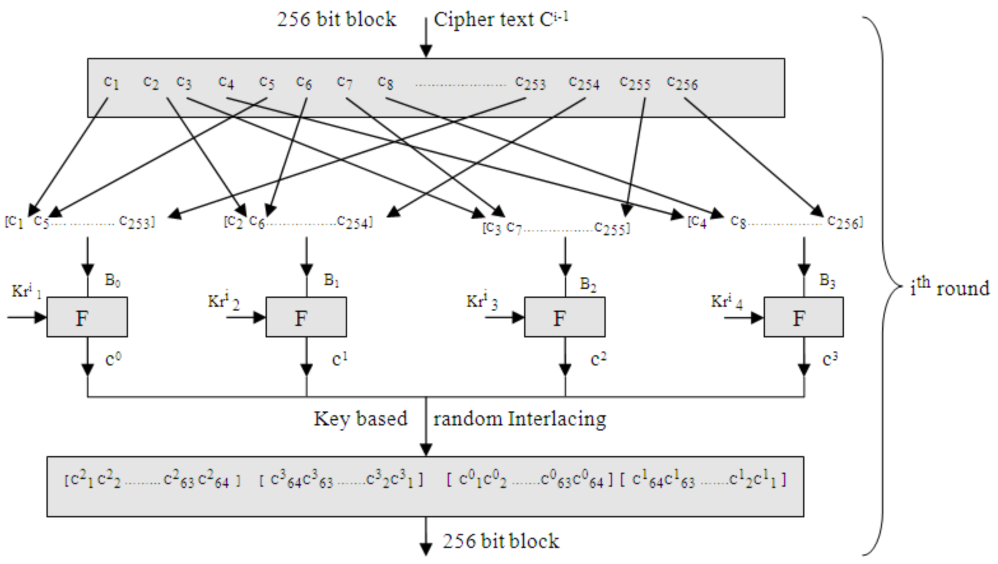
256 bit block
Cipher text C^{i-1}
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 .... c_{253} c_{254} c_{255} c_{256}
$[c_1\ c_5 \ldots c_{253}]$
$[c_2\ c_6 \ldots c_{254}]$
$[c_3\ c_7 \ldots c_{255}]$
$[c_4\ c_8 \ldots c_{256}]$
Kr^i_1 Kr^i_2 Kr^i_3 Kr^i_4
B_0 B_1 B_2 B_3
F
c^0 c^1 c^2 c^3
Key based random Interlacing
$[c^2_1 c^2_2 \ldots c^2_{63} c^2_{64}]$ $[c^3_{64} c^3_{63} \ldots c^3_2 c^3_1]$ $[c^0_1 c^0_2 \ldots c^0_{63} c^0_{64}]$ $[c^1_{64} c^1_{63} \ldots c^1_2 c^1_1]$
256 bit block
i^{th} round

Fig. 2: Illustration of key based random interlacing and decomposition during decryption

Algorithm for Interlacing

```
> $C^i_j$ <
BEGIN

 for n = 1 to 64
 {
 $C^{i-1}$ [ ((n-1)*4) +1] = $C^i$j [ n ]
 }

END
```

Algorithm for Decomposition

```
< $C^i$ > // during $i^{th}$ round
BEGIN

    j = 0
    k = 1
    for n = 1 to 256
    {
        $B^i_j$ [k] = $C^i$[n]
        j = ( j + 1 ) mod 4
        if ( j = = 0 )
        k = k + 1
    }

END
```

Algorithm for key based random decomposition:

```
← $C^i$ → // during $i^{th}$ round
BEGIN

j = $d_i$
if ( ( $K_i$ mod 2 ) = = 0 )
{
order = 0
}
else
{
order = 1
}            // 1: R → L and 0: L → R order
x = 1
for m = 1 to 4
{
      if ( order = = 0 )
      {
        p = 1
          for n = x to x + 63
      {
         $B^{i-1}_j$ [p] = $C^{i-1}$[n]
         p = p + 1
      }
        j = ( j + 1 ) mod 4
      }
      else
      {
      p = 64
        for n = x to x + 63
            {
               $B^{i-1}_j$ [p] = $C^{i-1}$[n]
               p = p - 1
            }
```

```
    j = ( j + 1 ) mod 4
    }
x = x + 64
}

END
```

Algorithm for Key based random Interlacing:

```
→ C^{i -1} j ← // during i^th round
BEGIN
d_i = K_i mod 4
x = d_i
y = 1
  if ( ( K_i mod 2 ) = = 0 )
  {
  order = 0
  }
  else
  {
  order = 1
  }
     While ( x Not equal to j )
     {
       x = ( x + 1 ) mod 4
       order = ( order + 1 ) mod 2
       y = y + 64
     }
       If ( order = = 0 )
        {
          for n = 1 to 64
          {
             C^i [y] = c^{i -1}_j[n]
             y = y +1
          }
        }
       else
        {
          p = y + 63
          for n = 1 to 64
            {
               C^i [p] = c^{i -1}_j[n]
               p = p -1
            }
        }
END
```

Note: Transformations to generation the round keys, required permutations and substitutions for function ‘F’ during encryption and decryption are similar to the one we already published (Kumar and Kumar, 2008).

RESULTS

Consider the plaintext:

P = {The big brown fox swam in water}

Let the key K = {He drowned in it}.

Let the 8 bit binary representation of plaintext P be:

```
01010100011010000110010100100000011000100110 1
001
0110011100100000011000100111001001101111011 10
111011011100010000000110011001101111011111 00000
100000011100110111011101100001011011010010000
0011010010110111000100000011101110110000101 11
0100011001010111001000101110
```

Let the key k be:

```
010010000110010100100000011001000111001001101
111011101110110111001100101011001000010000001
10100101101110001000000110100101110100
```

Let $d_i = K_i \bmod 4$.

We get $d_0 = 0$, this indicates that key based random decomposition begins with B_0 in first round. As K_0 is an even number, the order for B_0 is from left to right, order for B_1 is from right to left, order for B_2 is from left to right and right to left for B_3.

As we use four different blocks B_0, B_1, B_2, B_3 of 64 bit each for encryption, we use algorithm 4.5 to get these four blocks (Fig. 1):

B^0_0 = {00000010100011011000000011110100111 0
001010001101110011111000000011}
B^0_1 = {11010001111100010100010111101011011
00110000101111101110000110110}
B^0_2 = {11011111110000000111011111001100101
010110 00100000001111111000111 10}
B^0_3 = {11111101100100000000111010100000100 10
1110101101110 0101111100011011}

Permute the bits in key ‘k’ by using the random key based permutations published in our previous study (Kumar and Kumar, 2008).

Let this permuted key be divided into four equal size blocks and used as round keys kr^1_1, kr^1_2, kr^1_3, kr^1_4. for blocks B^0_0, B^0_1, B^0_2, B^0_3.respectively.

Now, we encrypt these four blocks with their respective round keys and with the help of round function ‘F’. Key based random permutations and key based random substitutions used in round key are similar to the one we derived in our previous study published (Kumar and Sastry, 2009).

Let the corresponding cipher blocks obtained after first round be c^1_0, c^1_1, c^1_2, c^1_3:

c^1_0 = {0101010001101000011001010010000001100010 011010010110011100100000}
c^1_1 = {01100010011100100110111101110111011011100010000001100110011011111}
c^1_2 = {0111100000100000011100110111011101100001011011010010000001101001}
c^1_3 = {0110111000100000011101110110000101110100011001010111001000101110}

We get the 256 bit cipher block C^1 after first round by applying interlacing described in Fig. 1 and 3:

C^1 = {0000111101111010001110010101000000001100111101001000000001000000000011111111001101001101011111110000001111111011000000110011001110000111111110001010001011100001000000101111110000101000110000101100001101111110001000011001101100000000011011110000011 1010101010110

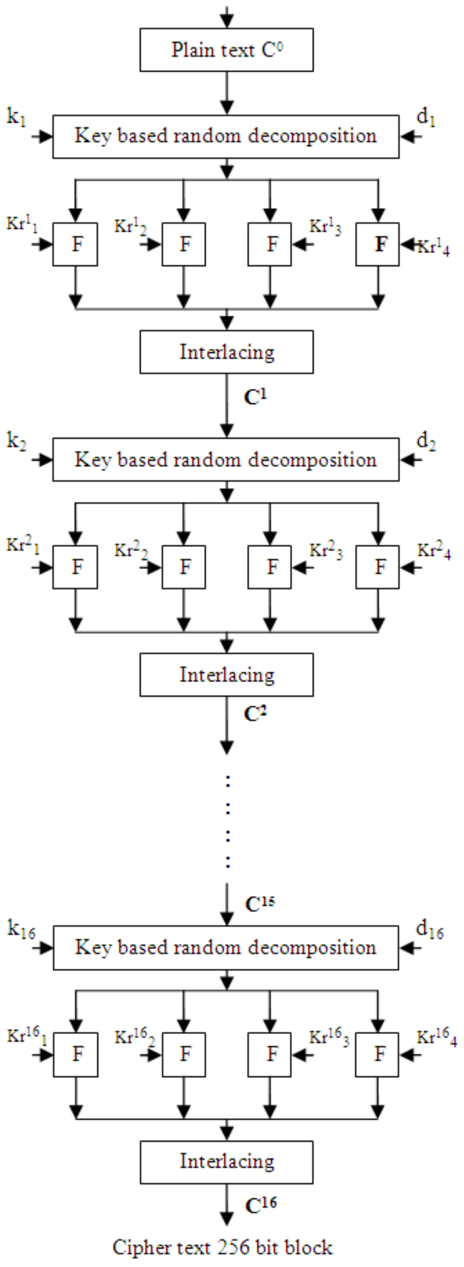

Fig. 3: The process of encryption

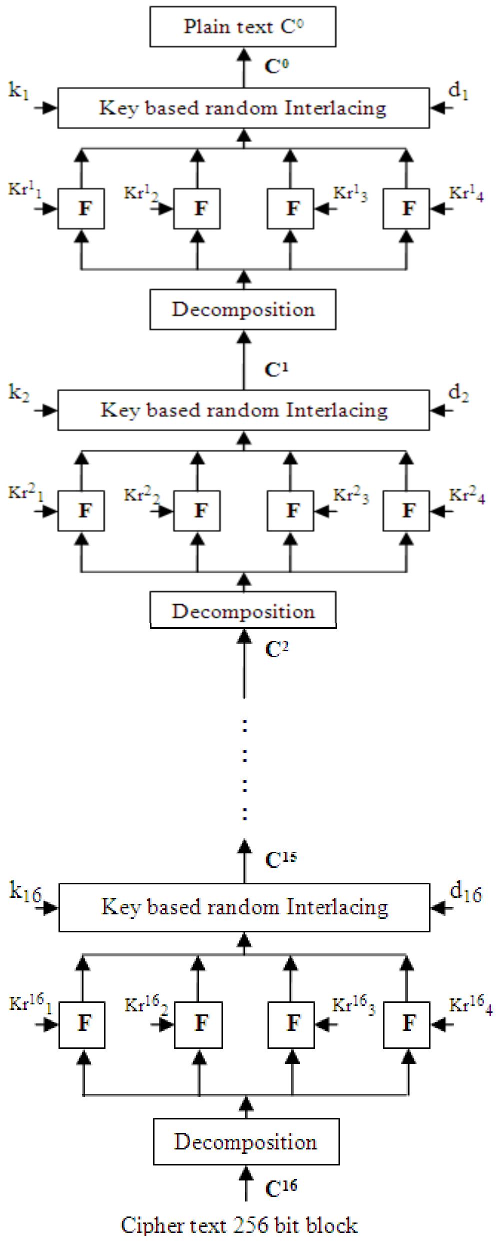

Fig. 4: The process of decryption

Similarly, by using the respective round and sub keys, we continue the encryption process up to 16 rounds and we get the final cipher as:

C^{16} = {000000101000110110000001111010011100010100011011100111110000001101101100001110111110
1000011001101101011110100010100011111000101111
01111111000000011101111100110010101011000100000
0011111100011110110110001111101001110110101110
10010000101011100000001001101111111}

In order to decrypt the cipher text, use the decryption algorithm; follow the transformations described in Fig. 2 for sixteen rounds. Thus, we get the required plaintext.

DISCUSSION

Cryptanalysis: To asses the strength of our encryption-decryption algorithms, we first show that brute force attack is not possible on our algorithm, next we show that even the well known "known plaintext attack" cannot break our cipher, followed by an analysis for avalanche effect to prove the strength of the cipher.

Brute force attack: According to brute force attack, if key space is small, then one can test all possible combinations of keys on encryption-decryption algorithms in some amount of time which acceptable to break the cipher. Therefore, key space should be large enough so that testing of all possible key combinations will take lot of time which is not acceptable in breaking a cipher.

As we have used 128 bit key in each round, the key space is:

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

Let us assume testing of one key on a computer takes 1 nano second. Then testing of 10^{39} keys will take $[(10^{39})/(10^9 \times 60 \times 60 \times 24 \times 365)]$ years, which is equal to more than a centaury. Since one cannot spend so much time in breaking the cipher, brute force attack is not possible on our algorithm.

Known plaintext attack: According to known plaintext attack, if enough number of plaintext – cipher text pairs are available then, one can understand the transformation used in developing the cipher. Our classical feistel cipher is prone to known plaintext attack due to the linearity that exit in transformations during encryption. Since we have used random key based decomposition and interlacing before and after encryption respectively. In every consecutive round, we have restricted the bits to get into different random blocks basing upon the key and the round. Hence we have successfully introduced a high degree of nonlinearity in our algorithm due to which more confusion and diffusion is added. Thus, known plaintext attack is not possible on our algorithm as an attacker is unaware of the way bits are scattering in to different blocks in different rounds. In this study, we prove that the strength of the cipher is good when we use key based random decomposition, key based random interlacing, interlacing and decomposition in our algorithm.

Avalanche effect: According to avalanche effect, for a plaintext P if C1 is an equivalent cipher then by keeping the key constant, if there is one bit change in plaintext P and we get an equivalent cipher as C2. Then the strength of the good if C1 and C2 differ by around 50% of the bits. Similarly, the algorithm can even be tested for a one bit change in key.

Let the plaintext be:

P = {The big brown fox swam in water}

Let the key be:

K = {He drowned in it}

Then by following the process of encryption described in algorithm and Fig. 1. We get the following cipher after 16 rounds:

C^{16} = {000000101000110110000000111101001110001010001101110011111000000110110110000111011111010
0001100110110101111010001010001111100010111101111111000000011101111100110010101011000100
00001111110001111011011000111110100111011010111010010000101011100000001001101111111}

Now, Let us change the plaintext by one bit. This can be done by changing the first letter in plaintext from 'T' to 'S' as the ASCII values of 'T' and 'S' differ by one. By keeping the key as constant.

Let the cipher text obtained for this new plaintext after 16 rounds of encryption be:

C^{16} = {1111111110110010000101000011001001100001000
00101111000100101010100010001000110110110011
1101
101001100000001110111001110010100111010000111
0010011000101101111110110110000000000110110110
01111000111000011010101010110101010001111001
111001101111101101 100010101110100}

From above two ciphers, we find that 134 bits differ out of 256 bits. Since around 50% of the bits differ in corresponding ciphers for a one bit change in plaintext; we say that strength of the cipher is good.

Now let us keep the plaintext as constant and change the key by one bit. This can be accomplished by changing the key character from ‘H’ to ‘I’ as their ASCII values differ by one bit. Let the corresponding cipher obtained after 16 rounds of encryption be:

C^{16} = {1110010011111110100101011011000000101001 0000
1111100111001000100100010101001110111000 11010
1101111000100111111001010111010001010100 01101
1101110010011100001001101111011011000010 10010
0111010110000111111010100110111000000000 01101
010111110000111001100110111110000}

In this case, we readily notice that 126 bits differ out of 256 bits. Therefore for a change in one bit in a key there is a difference of around 50% of bits in the corresponding ciphers. Thus the avalanche effect good for our ciphers when key based random decomposition, key based random interlacing, decomposition and interlacing is used in our encryption-decryption algorithms.

CONCLUSION

In conventional feistel cipher, we observed that known plaintext attack is possible because a set of bits will undergo into similar transformations and enter into same substitution box in each round. This makes the cryptanalysis work easy. In our recent research (Kumar and Kumar, 2008; Kumar and Sastry, 2009), we proved how “random key based permutations and substitutions” bring variable transformations in each round (Hussain and Ajilouni, 2006) In the present study, we have used a similar strategy “key based random interlacing and key based random decomposition” to strengthen the cipher further and to make the cryptanalysis more difficult. The results of avalanche effect seen indicates that the key based random interlacing and key based random decomposition introduced to counter attack the known plaintext attack provides good strength to the cipher.

ACKNOWLEDGMENT

The authors are very thankful to Prof. Depanwita Roy Chaudhury, IIT Kharagpur, India, for giving necessary suggestions and for her valuable inputs given while writing this study. The authors are very thankful to the management of Sreenidhi Institute of Science and Technology, for their support and encouragement given during this research work.

REFERENCES

Hussain, S.M. and N.M. Ajilouni, 2006. Key based random permutation. J. Comput. Sci., 2: 419-421. http://www.scipub.org/fulltext/jcs/jcs25419-421.pdf

Kumar, K.A. and S.U. Kumar, 2008. Block cipher using key based random permutations and key based random substitutions. Int. J. Comput. Sci. Network Security, 8: 267-277. http://paper.ijcsns.org/07_book/200803/20080339.pdf

Kumar, K.A. and V.U.K. Sastry, 2009. Modified Feistel cipher involving Interlacing and decomposition. Int. J. Comput. Sci. Network Security, 1: 77-82. http://ijcns.org/papers/Vol.1_No.2/091112.pdf

Stallings, W., 2003. Cryptography and Network Security: Principles and Practices. 3rd Edn., Cisco Press, ISBN: 1587050250, pp: 774.

Tavares S.E. and H.M. Heys, 1995. Avalanche characteristics of substitutions-permutation encryption networks. IEEE Trans. Comput., 44: 1131-1139. DOI: 10.1109/12.464391