Novel Method in Secure Data Communication Using MK15-Variable Key Cryptosystem

Krati Bajpai
TKR Engineering College, Meerpeth
Hyderabad, India

Manish Kr. Tiwari
Advanced Systems Laboratory,DRDO
Hyderabad, India

Abstract

In this paper, we are proposing a new encryption scheme MK15, influenced by both symmetric key and asymmetric key encryption methodologies. This scheme works as a public key cryptosystem, having both public key and private key and form the product cipher after transposition and substitution on plain text. The encryp-tion key will be drawn from decryption key. One magi-cal number is used for the generation of decryption key and this number is known to key generator only. This scheme is helpful in transferring data on an insecure communication channel. It is a variable size key based algorithm. The size of the key depends on the plain text.

Keywords: Asymmetric Key, Public Key, Private Key, Product Cipher, Variable Size Key

1 Introduction

Cryptography is a technique for secure data communi-cation on an insecure communication channel. Cryp-tography is a practice and techniques for hiding the in-formation as well as secure data communication in the presence of third parties. The main goal of Cryptogra-phy is data confidentiality, data integrity, authentication, and non-repudiation.

Cryptography algorithms are split into two categories: Symmetric key Cryptosystem and Public key Cryptosystem. In the Symmetric key Cryptosystem, sender and receiver share a common private key, this same private key is used for encryption and decryption of the message. A significant disadvantage of Symmetric key Cryptosystem is the key management necessary to use them securely. While in the Public key Cryp-tosystem, also known as Asymmetric key Cryptosys-tem, two keys being used between sender and receiver in which public key is known to all and private key is known to receiver only. The main advantage of Public key Cryptosystem is that even if an eavesdropper hears the transmitted message, he can not make any sense from the message until he do not know how to decrypt it.

The RSA Cryptosystem is a public key system and it is widely used for data communication. The modular exponentiation is one of the most important operation of RSA. The security of RSA Cryptosystem lies in intractability of factoring a large integer into some component primes. A very major threat to RSA would be a solution to the Riemann hypothesis. If a solution of this hypothesis were found, it would become very easy to find prime numbers.

The first symmetric key encryption standard,DES (Data Encryption Standard), was recommended by NIST (National Institute of Standards and Technology). DES applies a 56-bit key to each 64-bit block of data. The process involves 16 rounds for encryption and decryption. In DES,encryption and decryption uses the same feistel structure approach. Data can be recovered from cipher only by using exactly the same key used to encipher it. Now a days DES is considered to be inse-cure for many applications because the size of the key is too small and because of this Advanced Encryption Standard (AES) has superseded DES.

1.1 The Public Key Cryptosystem

In the symmetric cryptography, encryption and decryption key is the same. This type of cryptosystem can be easily broken if key is found. To avoid this problem and to improve protection mechanism Public Key Cryptosystem was introduced in 1976. There is an encryption procedure E to encrypt the plain text using encryption key i.e. to convert plain text into cipher text and decryption procedure D to decrypt the cipher text using decryption key i.e. to convert cipher text into original plain text.

In the public key system, the private key and public key has a mathematical relationship. There is always a possibility of achieving a private key by attacking on world-wide known public key. To avoid such problem make a cryptosystem in such a way that deriving private key from public key become impossible.

2 Review of Existing Literature

An implementation of the Diffie-Hellman key distribu-tion scheme that achieves a public key cryptosystem has

been discussed in [1]. A new type of cryptographic system, which minimize the need for secure key distribution channel has been discussed in the paper [2]. Imple-mentation of RSA algorithm with the help of Object-Oriented methods has been discussed in [3]. This pa-per [4] discussed about the performance improvement of RSA algorithm. In this paper, [5] attention has been given to quantum encryption. In this paper [6], vari-ous asymmetric algorithms like (RSA, Elliptic Curve, OAEP) are compared.

3 Proposed Algorithm

The proposed new scheme is an asymmetric block ci-pher. It is a parameterised algorithm, with a variable block size and sub block size, a variable number of rounds and a variable key length. The total size of the key will be equal to size of the plain text and this will be divided into parts which is equal to size of sub block. This flexibility provides an opportunity to en-hance both performance characteristics and information security level.

This particular MK15 algorithm is designated as MK15-B/b/r. The number of bytes in a block, B, is first parameter of MK15. Different choices of this pa-rameter result in different MK15 algorithm. The num-ber of bytes in a sub-block, b, is second parameter of MK15.The number of round will represented by, r, and this is the third parameter of MK15. MK15 is an iter-ative scheme because of, r, number of rounds. These parameters are summarised as follows:

SIZE_OF_BLOCK: The block size, in bytes.

SIZE_OF_SUB_BLOCK: The sub-block size, in bytes. This will be same as number of bytes in secret keys used for encryption and decryption.

r: The number of rounds.

N_B: Total number of blocks.

S_B: Total number of sub-blocks in a block.

L: The b-byte secret key will be: K[0][b], K[1][b],K[2][b],K[3][b],. . . ,K[N_B-1][b].

MK15 consists of three components: a key genera-tion algorithm, an encryption algorithm and a decryp-tion algorithm. These algorithms use the following two primitive operations:

1. : Bit-wise exclusive-OR.
2. n: Shifting symbol: the shift of x to the left by y bits is denoted by xny.

This procedure as in [7] also has the following three properties:

a) Deciphering the enciphered form of a message M yields M. Formally, D(E(M)) = M.

b) Both E and D are easy to compute.

c) By publicly revealing E the user does not reveal an easy way to compute D.

The most important feature of MK15 is its simplic-ity, which makes it easy to implement. Another impor-tant feature of MK15 is its data-independent key gener-ation.

3.1 Key Generation Algorithm

The most important part of any public key cryptosys-tem is to generate the encryption and decryption key and the way in which encryption key shared with the other users. The encryption or decryption procedures typi-cally consist of a method and encryption or decryption key. This method uses the encryption key to encipher the message into cipher text and decryption key to de-cipher it. The encryption and decryption key of MK15 scheme has following features:

1. Size of the encryption and decryption key will be of variable size and depends on the input text given by the key generator. The size of the key will be equal to size of plain text, which will be divided into size of sub blocks.
2. The encryption key will be generated with the help of Decryption key.
3. For the key generation, two large numbers will be used to generate two large prime numbers.
4. The deterministic random number generator (DRNG) algorithm will be used to generate a large random number with the help of large prime num-bers. The session key(third large number) will be provided as a seed value for this DRNG.
5. Two large prime numbers will be generated with the help of these two large numbers and the value of these prime numbers will be greater than the in-put numbers.

3.1.1 First algorithmic step for Key Generation:

1. Choose Input1, Input 2, Input 3, Input 4 and Input 5 any large numbers.
2. Calculate total number of characters in the Plain Text.
3. Divide the total size of Plain text into equal number of parts, the size of each part will be less than or equal to SIZE OF BLOCK.
4. Choose a Magical Number. The value of this Magical number should be less than or equal to SIZE_OF_SUB_BLOCK.

Algorithm 1 INPUT_AND_SIZE_EXTRACTOR

Input: Total number of characters in Plain Text, Input 1, Input 2, Input 3, Input 4, Input 5 and Input 6. Output: Encryption and Decryption Keys

```
1: SESSION KEY    input1
2: RAND NU M 1    input2
3: RAND NU M 2    input3
4: SIZE OF BLOCK    input4
5: SIZE OF SU B BLOCK    input5
6: MAGICAL NU M    input6
7: total char    total char in plain text
8: while 1 do
9:    if total char > SIZE OF BLOCK then
10:      KEY GENERAT OR(SIZE OF BLOCK;
         SIZE OF SU B BLOCK; SESSION
         KEY; RAND NU M 1; RAND NU M 2)
11:   else
12:      KEY GENERAT OR(total char;
         SIZE OF SU B BLOCK; SESSION
         KEY; RAND NU M 1; RAND NU M 2)
13:   end if
14:   total char    total char    SIZE OF BLOCK
15: end while
```

3.1.2 Second algorithmic step for Key Generation:

1. This step is to generate two ^{0}two dimensional^0 arrays, one for encryption key and other for de-cryption key.

2. The size of encryption key and decryption key will be ^{0}Key Index sizeo f (sub block)^0.

3. Generate two prime numbers X¶ and Y¶ and generate a large random number R using DRNG with the help of these two prime numbers and seed value Z.

$$\text{Key\_Index} = {}^{f}\ B=b+1 \quad i\ f\ B\%b! = 0 \quad (1)$$

$$B=b \quad i\ f\ B\%b = 0 \quad (2)$$

$$RC=b+1 \quad i\ f\ RC\%b! = 0\ \&\&\ \text{sizeo f (RC)} < B \quad (3)$$

$$RC=b \quad i\ f\ RC\%b = 0\ \&\&\ \text{sizeo f (RC)} < Bg \quad (4)$$

Where B is SIZE OF BLOCK, b is SIZE_OF_SUB_BLOCK and RC is remaining characters. The value of remaining characters will be less than SIZE OF BLOCK.

Algorithm 2 KEY_GENERATOR

Input: Argument 1, Argument 2, Argument 3, Argu-ment 4 and Argument 5.
Output: Encryption and Decryption keys.

```
1: if Arg1%Arg2! = 0 then
2:    key index    Arg1=Arg2 + 1
3:    loop variant    key index 1
4:    odd data 1
5: else
6:    key index    Arg1=Arg2
7:    loop variant    key index
8:    odd data 0
9: end if
10: RAND NU M    RAND NU M 1 + RAND NU M 2
11: Encryption Key File    R
12: Decryption Key File    RAND NU M 1
13: Decryption Key File    RAND NU M 2
14: for k 0 to loop variant do
15:    for i 0 to SIZE OF SU B BLOCK do
16:       key value
       RAND NU M GENERAT OR(Arg3; Arg4; Arg5)
          % SIZE OF SU B BLOCK
17:       for j    to i do
18:          if D[k; j] = key value then
19:             break
20:          end if
21:       end for
22:       if j = i then
23:          D[k; i]    key value
24:          temp    D[k; i]
          (MAGICAL NU M%SIZE OF SU B BLOCK)
25:          Decryption Key File    temp
26:          i    i + 1
27:       end if
28:    end for
29: end for
30: for i 0 to loop variant do
31:    for j    0 to SIZE OF SU B BLOCK do
32:       index    D[ j; i]
          (MAGICAL NU M%SIZE OF SU B BLOCK)
33:       E[ j; index]    i index
34:    end for
35: end for
36: for i0 to loop variant do
37:    for j    0 to SIZE OF SU B BLOCK do
38:       Encryption Key File    E[i; j]
39:    end for
40: end for
```

```
41: if odd_data = 1 then
42:   for i  0 to (Arg1%SIZE_OF_SU B_BLOCK) do
43:     key_value
        RAND_NU M_GENERAT OR(Arg3; Arg4; Arg5)
        % SIZE_OF_SU B BLOCK
44:     for  j    to i do
45:       if  D[key_index  1; j] = key_value then
46:         break
47:       end if
48:     end for
49:     if j = i then
50:       D[key_index  1; i]    key_value
51:       temp    D[key_index  1; i]
        (MAGICAL NU M%SIZE OF_SU B BLOCK) 11:
52:       Decryption_Key_File    temp
53:       i    i + 1
54:     end if
55:   end for
56: end if
57: if  odd_data = 1 then
58:   for i    0 to (Arg1%SIZE OF_ SU B BLOCK) do
59:     index    D[key_index  1; i]
        (MAGICAL NU M%SIZE OF_SU B BLOCK) 20:
60:     E[key_index  1; index]    i  index
61:   end for
62: end if
63: if  odd_data = 1 then
64:   for i    0 to (Arg1%SIZE_ OF_ SU B BLOCK) do
65:     Decryption_Key_File    E[key_index  1; i]
66:   end for
67: end if
```

Algorithm 3 RAND_NUM_GENERATOR

Input: Three large numbers X, Y, Z.
Output: One random number Key Value.

```
1: R    X +Y
2: start    Z
3: X¶        PRIME_NU M_GENERAT OR(X;Y )   :
   where X¶ > X
4: Y¶        PRIME_NU M_GENERAT OR(X;Y )   :
   where Y¶ > Y
5: start    value  X¶
6: start    value +Y¶
7: result   (start=2^ 32)mod(2^ 20)
8: result n12
9: Interchange Y¶ and X¶
10: Z    result
11: key_valueresult
12: Repeat (a) to (e) steps 3 times
13: return key_value
```

Algorithm 4 PRIME_NUM_GENERATOR

Input: Two large random numbers X and Y.
Output: Two large prime numbers X¶ and Y¶.

```
1: while 1 do
2:   for j    1 to X do
3:     if X% j = 0 then
4:       break
5:     end if
6:   end for
7:   if j = X then
8:     break
9:   end if
10:  X    X + 1
   end while
12: X¶    X
13: while 1 do
14:   for j    1 to Y do
15:     if Y % j = 0 then
16:       break
17:     end if
18:   end for
19:   if j = Y then
        break
21:   end if
22:   Y    Y + 1
23: end while
24: Y¶    Y
25: return X¶, Y¶
```

In the two dimensional array for Encryption and Decryption key, Key_Index will be equal to number of rows and b (size of sub block) will be equal to number of columns.

The public key becomes < B; b; R; E_1; E_2; E_3; : : : ; E_n > where n will depend on size of block and sub-block. The private key becomes < B; b; R; D_1; D_2; D_3; : : : ; D_n > where n will depend on size of block and sub-block. The size of E_1; E_2; : : : ; E_n and D_1; D_2; D_3; : : : ; D_n will be equal to size of sub-block.

3.2 Encryption Algorithm

Suppose Tom wants to send a message M to Jerry. To encrypt the message using the proposed algorithm scheme, Tom must obtain encryption key of Jerry which is public and known to all. The message to send must now be encrypted with this encryption key. The message M must be represented in the range of [0-size of block] and the size of the block will be same as size of block of key.

3.2.1 First algorithmic step for Encryption:

1. Calculate total number of characters, L, in the plain text M.

2. Divide the plain Text M into L/B number of blocks where L is the size of plain text and B is the size of each blocks.
3. Total number of blocks

$$N_B = fL=B; \quad i\,f\ L\%B = 0 \qquad (5)$$

$$L=B + 1; \quad i\,f\ L\%B! = 0g \qquad (6)$$

4. The total number of blocks in case of L%B! = 0 will be L=B + 1, where the size of last block will not be equal to B whereas rest of the blocks will be of size B.
5. Further divide each block into sub blocks of size, b.
6. Total number of sub blocks of size, b, will be

$$S_B = fB=b; \quad i\,f\ B\%b = 0 \qquad (7)$$

$$B=b + 1; \quad i\,f\ B\%b! = 0g \qquad (8)$$

7. The total number of sub-blocks in case of B%b! = 0 will be B=b + 1, where the size of last sub-block will not be equal to b whereas rest of the sub-blocks will be of size b.

3.2.2 Second algorithmic step for Encryption:

1. The encryption key will be extracted from Encryp-tion key file.
2. This step is to fill ^{0}two dimensional^0 array for en-cryption key.
3. The size of encryption key will be ^{0}Key Index sizeo f (sub block)^0.

$$Key\_Index = fB=b + 1 \quad i\,f\ B\%b! = 0 \qquad (9)$$

$$B=b \quad i\,f\ B\%b = 0 \qquad (10)$$

$$RC=b + 1 \quad i\,f\ RC\%b! = 0 \ \&\&\ sizeo\,f\,(RC) < B \qquad (11)$$

$$RC=b \quad i\,f\ RC\%b = 0 \ \&\&\ sizeo\,f\,(RC) < Bg \qquad (12)$$

Where RC is Remaining characters. The value of remaining characters will be less than size of Block.

3.2.3 Third algorithmic step for Encryption:

a) Divide the plain Text M into l = n=k number of packets where n is the size of plain text, k is the size of each packet and l is total number of packets.

b) Apply the permutation on each packet based on the encryption key $M_1[u]$ $M_1[u] + E_1$, Where $0 < u < l$.

c) $M_1[u]$ $M_1[u]$ Z. (Where Z is a large random value)

d) Step b will take place for rest of the packets.

Algorithm 5 INPUT AND SIZE EXTRACTOR ENCRYPTION

Input: Total number of characters in Plain Text,
Input 1, Input 2, Input 3, Input 4, Input 5 and Input 6.
Output: Encryption and Decryption Keys

```
1: RAND_NU M   Encryption Key File
2: SIZE_OF_BLOCK   Encryption Key File
3: SIZE_OF_SU B_BLOCK   Encryption_Key_File
4: total_char   total_char in plain text
5: while 1 do
6:   if total_char > SIZE_OF_BLOCK then
7:     DATA_EX T RACT OR(SIZE OF BLOCK;
       SIZE_OF_SU B_BLOCK; RAND NU M 1)
8:   else
9:     DATA_EX T RACT OR(total char;
       SIZE_OF_SU B_BLOCK; SESSION KEY;
10: RAND_NU M)
11:  end if
12:  total_char   total_char   SIZE_OF_BLOCK
13: end while
```

Algorithm 6 DATA_EXTRACTOR ENCRYPTION

Input: Key_Index.
Output: Encrypted Message.

```
1: if Argument1%Argument2! = 0 then
2:   key_index   Argument1=Argument2 + 1
3:   loop_variant   key_index  1
4:   odd_data  1
5: else
6:   key_index   Argument1=Argument2
7:   loop_variant   key_index
8:   odd_data  0
9: end if
10: for k  0 to loop variant do
11:   for i  0 to SIZE OF SU B BLOCK do
12:     Data[i; j]
        read_input_plain text f ile(char)
13:     Key[i; j]
        read_encryption_key f ile(int)
14:   end for
15:   DATA_ENCRY PT ION(data; Key; SIZE OF SU B BLOCK)
16: end for
17: if odd data = 1 then
18:   for i  0 to total char%SIZE_OF SU B_BLOCK
    do
19:     Data[key_index  1; i]
        read_input_plain text f ile(char)
20:     Key[key_index; i]
        read_encryption_key f ile(int)
21:   end for
22:   DATA_ENCRY PT ION(total char%SIZE OF SU B BLOCK; Key;
        SIZE_OF_SU B_BLOCK)
23: end if
```

e) In the last merge all the packets and this will be-come cipher Text C.

Algorithm 7 DATA_ENCRYPTION

Input: Plain text, Encryption Key, length and random value.

Output: Encrypted Message.

```
for i 1 to length do
  Data_Encrypt[i] 0
end for
for i 1 to length do
  index Key[i]
  index index i
  Data_Encrypt[i] data[index]
  Data_Encrypt[i] Data_Encrypt[i] i
  Data_Encrypt[i] Data_Encrypt[i]
    (RAND NU M)mod(256)
end for
for i 1 to length do
  cipher_text_f ile Data_Encrypt[i]
end for
```

3.3 Decryption Algorithm

Jerry has received encrypted message send by Tom af-ter encrypting with Jerrys encryption key. Jerry will de-crypt the message using own decryption key, which is known to Jerry only. Jerry will apply same procedure as Tom has applied for encryption:

3.3.1 First algorithmic step for Decryption:

1. Calculate total number of characters, L, in the plain text M.
2. Divide the plain Text M into L/B number of blocks where L is the size of plain text and B is the size of each blocks.
3. Total number of blocks

$$N_B = fL=B; \quad i f\ L\%B = 0 \qquad (13)$$

$$L=B + 1; \quad i f\ L\%B! = 0g \qquad (14)$$

The total number of blocks in case of L%B! = 0 will be L=B + 1, where the size of last block will not be equal to B whereas rest of the blocks will be of size B.

4. Further divide each block into sub blocks of size, b.
5. Total number of sub blocks of size, b, will be

$$S_B = fB=b; \quad i f\ B\%b = 0 \qquad (15)$$

$$B=b + 1; \quad i f\ B\%b! = 0g \qquad (16)$$

The total number of sub-blocks in case of B%b! = 0 will be B=b + 1, where the size of last sub-block will not be equal to b whereas rest of the sub-blocks will be of size b.

3.3.2 Second algorithmic step for Decryption:

1. The decryption key will be extracted from Decryp-tion key file.
2. This step is to fill ^{0}two dimensional^0 array for de-cryption key.
3. The size of decryption key will be ^{0}Key Index sizeo f (sub block)^0.

$$Key\ Index = fB=b + 1 \quad i f\ B\%b! = 0 \qquad (17)$$

$$B=b \quad i f\ B\%b = 0 \qquad (18)$$

$$RC=b + 1 \quad i f\ RC\%b! = 0 \&\& sizeo f\ (RC) < B \qquad (19)$$

$$RC=b \quad i f\ RC\%b = 0 \&\& sizeo f\ (RC) < Bg \qquad (20)$$

Where RC is Remaining characters. The value of remaining characters will be less than size of Block.

3.3.3 Third algorithmic step for Decryption:

a) Divide the plain Text M into l = n/k number of packets where n is the size of plain text, k is the size of each packet and l is total number of packets.

b) $C_1[u]$ $C_1[u]$ Z. (Where Z is a large random value)

c) Apply the permutation on each packet based on the encryption key $C_1[u]$ $C_1[u]+D_1$, Where $0 < u < l$.

d) Step b will take place for rest of the packets.

e) In the last merge all the packets and this will be-come Plain Text.

Algorithm 8 INPUT AND SIZE EXTRACTOR DECRYPTION

Input: Total number of characters in Plain Text,

Input 1, Input 2, Input 3, Input 4, Input 5 and Input 6. Output: Encryption and Decryption Keys

```
RAND_NU M_1 Decryption_Key_File
RAND_NU M_2 Decryption_Key_File
SIZE_OF_BLOCK Decryption_Key_File
SIZE_OF_SU B_BLOCK Decryption_Key_File
total_char total_char _in_plain_text
RAND_NU M RAND_NU M_1 + RAND_NU M_2
while 1 do
  if total_char > SIZE_OF_BLOCK then
    DATA_EX T RACT OR(SIZE_OF_BLOCK;
    SIZE_OF_SU B_BLOCK; RAND_NU M_1)
  else
    DATA_EX T RACT OR(total char;
    SIZE_OF_SU B_BLOCK; SESSION KEY;
RAND_NU M)
  end if
  total_char total_char SIZE_OF_BLOCK
end while
```

Algorithm 9 DATA_EXTRACTOR DECRYPTION

Input: Key_Index.
Output: Encrypted Message.

```
1: if Argument1%Argument2! = 0 then
2:    key_index    Argument1=Argument2 + 1
3:    loop_variant    key_index  1
4:    odd_data   1
5: else
6:    key_index    Argument1=Argument2
7:    loop_variant    key_index
8:    odd_data   0
9: end if
10: for k   0 to loop_variant do
11:    for i   0 to SIZE_OF_SU B_BLOCK do
12:       Data[i; j]
          read_input_plain_text_f ile(char)
13:       Key[i; j]
          read_decryption_key_f ile(int)
14:    end for
15:    DATA_DECRY PT ION(data; Key;
          SIZE_OF_SU B_BLOCK)
16: end for
17: if odd_data = 1 then
18:    for i 0 to total_char%SIZE_OF_SU B_BLOCK
    do
19:       Data[key_index  1; i]
          read_input_plain_text_f ile(char)
20:       Key[key_index; i]
          read_decryption_key_f ile(int)
21:    end for
22:    DATA_DECRY PT ION(total_char%
          SIZE_OF_SU B_BLOCK; Key;
          SIZE_OF_SU B_BLOCK)
23: end if
```

4 Experimental Results

The table given below will give idea about the ef-ficiency of the MK15 algorithm. It gives the detail about the time(in second) taken by the algorithm for Key Generation, Encryption and Decryption, where the size of the Message is 4KB. The time taken by the Key Generation algorithm will be more if the size of block and sub block increases but this will also increase the complexity and security of the key. The size of the Encryption and Decryption key will be equal to size of Message. The time taken by Encryption and Decryption algorithm will remain the same in all the cases. The main benefit of this algorithm is that it is controlled by the receiver. Whenever sender wants to send a message, she has to communicate to the receiver about the size of the message. Then receiver will generate a dummy message of the same size and generate the keys with the help of the dummy message.

Algorithm 10 DATA_DECRYPTION

Input: Plain text, Decryption Key, length and random value.
Output: Decrypted Message.

```
1: for i   1 to length do
2:    Data_Decrypt[i]    0
3: end for
4: for i   1 to length do
5:    index   Key[i]
6:    index   index  i
7:    Data_Decrypt[i]    data[index]
8:    Data_Decrypt[i]    Data_Decrypt[i]  i
9:    Data_Decrypt[i]    Data_Decrypt[i]
        (RAND NU M)mod(256)
10: end for
11: for i   1 to length do
12:    cipher_text_f ile    Data_Decrypt[i]
13: end for
```

The receiver will send encryption key to the sender and keep decryption key with her. The sender will encrypt the message with encryption key and receiver will decrypt it with the help of decryption key.In the example given below, the size of Block and Sub-Block is ginev in bytes while time is in ms.

Table 1:
Summary of characteristics of the programs under test

Block Size	Sub-block Size	Key Gen.Time	Encrypt.Time	Decrypt.Time
8	2	0.846	0.003	0.003
16	4	1.539	0.003	0.003
32	8	2.607	0.003	0.003
64	16	4.555	0.003	0.003
256	32	7.457	0.003	0.003
512	64	9.337	0.003	0.003
768	96	11.670	0.003	0.003
1024	128	14.383	0.003	0.003
1536	192	17.909	0.003	0.003
1792	224	19.340	0.003	0.003

Example:

Plain Text: ”Secure Communication Using MK15”
Encryption Key: 128 64 1 -2033179352 5 18 8 10 13 21 9 29 31 25 16 7 22 33 12 19 11 30 14 6 32 17 28 0 3 24 2 23 15 1 26 4 27 20
Decryption Key: 128 64 1 34567 6543223 28 24 27

27 5 21 12 10 15 9 27 2 9 28 19 26 4 19 28 53 16 26 12
1 16 4 59 10 26 15 23 52 44
Cipher Text: ScOKN Y kQT NOK My
HY OLDzUY R
Size of Plain Text: 34 bytes
Size of Cipher Text: 34 bytes
Key Generation Time: 0.072 Second
Encryption Time: 0.001 Second
Decryption Time: 0.001 Second

System Details:

Model Name: Intel(R) Core(TM) 2 Duo CPU @ 3.00 GHz
Cache Size: 4096 KB
Operating System: RHEL 5.3

5 Conclusions

The proposed method for secure communication is based on asymmetric or public-key cryptosystem. The security of this algorithm rests in the part of difficulty of finding decryption key with the help of encryption key. The method permits a secure communication without the use of couriers to carry keys. The different pattern of keys can be generated by changing the size of block, sub-block, random numbers and session key. The strik-ing feature of MK15 encryption algorithm is that for same input plaintext, the cipher text generated each time will be different. The advantage of different keys gener-ated for the same input is that it will greatly enhance the security aspect of the algorithm. The receiver can send different keys to the sender depending on size of block, sub-block, random numbers and session key having the size of plain text same all the time.

Future Work

MK15 algorithm is very useful when plain text is small in size. In the case of bigger files(more than 5KB), this method takes a lot of time but the message will be highly secure. To avoid this time problem, there is need of an algorithm which has fixed size encryption and decryption key as well as it should be independent of plain text. This algorithm should encrypt and de-crypt the message in minimum amount of time. MK15 algorithm is useful in case of Encryption only, in fu-ture improvements is required so than it can be used for both Encryption and Digital Signature. Comparison of MK15 public key cryptosystem with other asymmetric and cryptosystem is required, which can be possible in future.

References

[1] Taher Elgamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions On Information Theory, Vol. IT-31, No. 4, July 1985.

[2] W. Diffie and M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, Vol. IT-22, pp. 472-492, 1976.

[3] Guicheng Shen, Bingwu Liu, Xuefeng Zheng, Re-search on Fast Implementation of RSA with JAVA, Proceedings of the 2009 International Symposium on Web Information and Application, May 22-24, 2009, pp.186-189.

[4] Qing LIU, Yunfei LI, Tong LI, Lin HAO, The Research of the Batch RSA Decryption Perfor-mance, Journal of Computational Information Sys-tems 7:3(2011) 948-955.

[5] T. Morkel, JHP Eloff, ENCRYPTION TECHNIQUES: A TIMELINE APPROACH, Information and Computer Security Architecture (ICSA).

[6] Neha Garg, Pratibha Yadav, Comparison of Asymmetric Algorithms in Cryptography, International Journal of Computer Science and Mobile Computing, Vol. 3, Issue 4, April-2014, pp. 1190-1196.

[7] R.L.Rivest, A.Shamir and L.Adleman, A Method for Obtaining Digital Signature and Public Key Cryptosystems, Communication of the ACM, Vol. 21, pp. 120-126, 1978.

[8] Williams Stallings, Cryptography and Network Se-curity, Prentice Hall, 6th Edition, 2013.

[9] Bruice Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C, Wiley India Pri-vate Limited, 2nd Edition, 2007.

[10] Hoffstein, An Introduction To Mathematical Cryp-tography, Springer International, 2011.

[11] https://en.wikipedia.org/wiki/Cryptography.

[12] https://en.wikipedia.org/wiki/Public-key_cryptography.

[13] https://en.wikipedia.org/wiki/Symmetric-key_algorithm.

[14] www.cacr.math.uwaterloo.ca/hac/.

[15] http://ciphersbyritter.com/GLOSSARY.HTM.

[16] http://www.cryptocorner.com/.