International Journal of Computer Applications (0975 – 8887)
Volume 70– No.18, May 2013

A Two Round Block Cipher Symmetric Key Cryptography based on Key Stream

Gautam Rakshit
National Institute of Technology
Agartala,India

Suman deb
National Institute of Technology
Agartala,India

Ashim Saha
National Institute of Technology
Agartala,Indi

ABSTRACT

Data security is one of the challenges of all times. In the modern computer parlance it is important to secure the vital data. One of the common aspects of data security is the privacy for which Cryptography is the promising methodology. Cryptographic computations can be used in different concerned areas of computer communication for encrypting and transmitting the information. cryptography has shown its effectiveness in the field of secured data transmission and much research work is going on to make the computational process more complex to the unauthorized users so that they cannot decrypt the information in a reasonable time. In this research work, strong substitution based encryption algorithms are proposed and the encryption and decryption process are broadly divided into two rounds and key stream generating procedures are proposed. The proposed procedures are being implemented, analyzed and it shows its efficiency in computation, storage and transmission; and it is more powerful during the decryption process. Proper care has been taken so as to keep the cipher text output file size as close to the plain text file size for fast transmission and besides these the decryption time is more than the encryption time. This paper includes the procedures like Substituent list generation, key generation, encryption and decryption that are continued and enhanced from the previous proposed work [10]. The performances are finally demonstrated and its implementations (using C # language) are explained and analyzed.

General Terms

Security, cryptography, cryptanalysis, encryption, decryption, key generation.

Keywords

Security, encryption, decryption, key generation, cipher text, plain, cryptosystem

1. INTRODUCTION

The word cryptography comes from two Greek words meaning “secret writing” and is the art and science of concealing meaning.Crytanalysis is the breaking of codes. The basic component of the cryptography is the cryptosystem.

A private-key encryption scheme [1], [2], [3] is comprised of three algorithms: the first is a procedure for generating keys, the second a procedure for encrypting, and the third a procedure for decrypting. These algorithms have the following functionality:

A. The key-generation algorithm Gen is a probabilistic algorithm that outputs a key K chosen according to some distribution that is determined by the scheme.

B. The encryption algorithm Enc takes as input a key K and a plaintext m and outputs a cipher text C. Encryption of the plaintext m using the key K by Enck (m).

D. The decryption algorithm Dec takes as input a key K and a cipher text can d outputs a plaintext m. Decryption of the cipher text c using the key K by Deck(c).

The procedure for generating keys defines a key space K (i.e., the set of all possible keys), and the encryption scheme is defined over some set of possible plaintext messages denoted M and called the plaintext (or message) space.

Since any cipher text is obtained by encrypting some plaintext under some key K and M define a set of all possible cipher texts that we denote by C. Note that an encryption scheme is fully defined by specifying the three algorithms (Gen; Enc; Dec) and the plaintext space M. The basic correctness requirement of any encryption scheme is that for every key K output by Gen and every plaintext message m 2M, it holds that Deck(Enck(m)) = m:

In other words, an encryption scheme must have the property that decrypting a cipher text (with the appropriate key) yields the original message that was encrypted. Recapping our earlier discussion, an encryption scheme would be used by two parties who wish to communicate as follows. First, Gen is run to obtain a key K that the parties share. When one party wants to send a plaintext m to the other, he would compute c: = Enck(m) and send the resulting cipher text c over the public channel to the other party. Upon receiving c, the other party computes m: = Deck(c) to recover the original plaintext.

In a nutshell, mathematically a cryptosystem is a 5-tuple(P,K,C,E,D), where P is the set of plain-texts, K is a set of keys, c is the set of cipher texts, E:P x K $\longrightarrow$ *C is the set of enciphering functions, and D:C x K* $\longrightarrow$*P is the set of deciphering functions.*

In the previous work as noted in [10] simulation has been done based on the DNA cryptography principles [4], [5], [6], [7], [8]. It has encryption/decryption algorithm where the generated cipher text size is more than twice the size of the original message which ultimately incurs a lot of network bandwidth. This was one of the pitfalls of the previous work.

In this research work the main objectives are to minimize the cipher size, to make encryption /decryption procedures more fast as compared to [10], increase decryption time compared to encryption time by making proper modification in the algorithms and to increase the key strength by introducing the key stream technique which allows 16 different keys to be applied to each byte of a input block. The proposed encryption algorithm works on block of input i.e. it generates block cipher.

2. PROPOSED SYMMETRIC KEY CRYPTOGRAPHY BASED ON STRONG SUBSTITUTIONS

The proposed symmetric key cryptography works on block cipher .It works on a block of 32 bytes (256 bits). The algorithm is divided into two broad rounds totally different from each other. In short the cryptosystem is as follows

Step-1: Generate Key Stream for second round using keys K2 and K3

Step-2: Generate Substituent list using key K1.

Step-3: Encrypt the plain text using the K3, K4, Key Stream and Substituent list.

The reverse think happens in decryption.

2.1 Sender's side Computations:

2.1.1 Procedure to generate Key Stream for second round

Input: 1) Two keys K2 (Range 0 to 255) and K3

2) Intermediate Byte array KeyStream [] of size 16.

Output: Filling the intermediate array KeyStream [] with 16 byte values**.**These bytes will be used in round 2 operations.

Steps:

```
Step 1.Let KeyStream [0] = K2.
Step 2 Let Integer variable K=K3.
Step 3.Let Integer Variable R=0.
Step 4.Repeat through step 8 for I=1 to 15.
Step 5.R=K + KeyStream [I-1].
Step 6.If (R>256)
Step 7.R=R%256.
Step 8.KeyStream[i] = Convert To Byte(R).
Step 9.Repeat through step15 for I=0 to 3.
Step10.Byte Variable P= Convert to Byte ((KeyStream[i*4] AND KeyStream [i*4+1])XOR ((NOT(KeyStream[i*4+2]))AND KeyStream [i*4+3]))
Step 11. Repeat through step 15 for J=0 to 3.
Step 12.R=KeyStream[i*4+j] + P.
Step 13.If (R>256)
Step 14.R=R%256.
Step 15.KeyStream [i*4+j]=Convert To Byte(R).
Step 16. Stop.
```

2.1.2 Procedure to prepare the Substituent list (array)

Input: 1) A byte array Substituent of size = 256.

2) Let Key K1 =14 (Range is 0 to 255).

Output: Filling the array with 256 different byte value using key1 .These bytes will be used for substitution purpose.

Steps:

```
Step 1.Let a Byte variable J= 0.
Step 2 Let Integer variable L=0.
Step 3.Repeat through Step 9 for I=0 to 255.
Step 4.L=I+K1;
Step 5.If (L<256) then Substituent [L] =J.
Step 6.Else Substituent [L-256] =J.
Step 7.End If
Step 8.J=J+1.
Step 9.I=I+1.
Step 10 Stop.
```

2.1.3 Procedure for Encryption

Input: i). Plain text.

ii). Let Key2=8(range is 5 to 255).

iii). Let Key3 =400(range is 255 to 999999999).

iv). Let Key4= 150 (KEY1 XOR KEY2 XOR (KEY3%256))

v). A List of auto generated Substituent using procedure B

vi). A List of Key Stream [] using procedure A which will supply different keys in Round 2

Output: A file containing the cipher text.

Steps:

```
Step 1.Select K2 number of random bytes from array Substituent and write to the cipher output file as the starting left padding.
Step 2 Read each block of plain text of size 16 bytes at a time in array Temp[] and repeat through step 14.
Round 1:
Step 3 Repeat through step 9 for I =0 to 15.
Step 4 ByteValue=Temp [I] +128(To ensure that negative bytes are also converted into positive values).
    E.g. ByteValue = 82+128=210.// Consider the first plain text byte be 82.
Step 5.Calculate q= (integer) ByteValue/16, e.g. q=210/16=13.
    And r = ByteValue %16, e.g r=210%16=2.
Step 6 Calculate index = r*16+q
    E.g. index=2*16+13=45.
  Using 'index' the algorithm will initiates another round by fetching the data at Substituent [index]
Step7. Let PreFinalIndex= Substituent [index] = -27 and add +128 again to ensure Positive PreFinalIndex = 101.
Step8. Calculate a new index; we name it as the 'FinalIndex' who's content will be the cipher byte as:
        i. Prefinalindx2 = (Prefinalindx + key3) % 256.
           e.g. Prefinalindx2 = (101 + 400) % 256=245.
        ii. FinalIndex = (Prefinalindx2 XOR KEY4) i.e. 99.
        iii.Cipherbyte= Substituent [Finalindex].
           E.g. Let Cipherbyte = 83.
Step9.Temp [I] = Cipherbyte.
        E.g. Let Temp [0] = 83.
```

ROUND 2:

```
Step 10 Repeat through Step 14 for J=0 to 15.
Step 11 If (J%2) ==0 Then
Step 12 Temp [J] = (Temp [J] + KeyStream [J]) %256.
        E.g. Let Temp [0] = (83 + 245) % 256=72.
Step 13 Else Temp [J] =Temp [J] XOR KeyStream [J].
Step 14 End If.
Step 15.Write Temp [] to output file.
Step16. Select K2 number of random bytes from array
Substituent and write to the output file for the ending
rightmost padding.
Step17. Stop
```

2.2 Receivers side computations

2.2.1 Key Stream generation for second round

Input: a.) Two keys K2 (Range 0 to 255) and K3.

b).Intermédiate Byte array KeyStream [] of size 16.

Output: Filling the intermédiate array KeyStream [16] with 16 byte valúes**.**These bytes Will be used in round 2 operations.

Steps:

```
Step 1.Let KeyStream [0] = K2.
Step 2 Let Integer variable K=K3.
Step 3.Let Integer Variable R=0.
Step 4.Repeat through step 8 for I=1 to 15.
Step 5.R=K + KeyStream [I-1].
Step 6.If (R>256)
Step 7.R=R%256.
Step 8.KeyStream[i] = Convert To Byte(R).
Step 9.Repeat through step15 for I=0 to 3.
Step10.Byte Variable P= Convert to Byte ((KeyStream[i*4]
AND KeyStream [i*4+1])XOR
((NOT(KeyStream[i*4+2]))AND KeyStream [i*4+3]))
Step 11 Repeat through step 15 for J=0 to 3.
Step 12.R=KeyStream[i*4+j] + P.
Step 13 If (R>256)
Step 14 R=R%256.
Step 15 KeyStream [i*4+j]=Convert To Byte(R).
Step 16 Stop.
```

2.2.2. Procedure to prepare the Substituent list (array):

Input: a. A byte array Substituent of size = 256.

b.Key1=14(Range is 0 to 255).

Output: Filling the array with 256 different byte value using key1 .These bytes will be used for substituting purpose.

Steps:

```
Step 1.Let a Byte variable J= -128.
Step 2 Let Integer variable L=0.
Step 3.Repeat through Step 9 for I=0 to 255.
Step 4.L=I+Key1;
Step 5.If (L<256) then Substituent [L] =J.
Step 6.Else Substituent [L-256] =J.
Step 7.J=J+1.
Step 8.I=I+1.
Step 9 Stop.
```

2.2.3. *Procedure for Decryption:*

Input: a. cipher text file.

b. Let K2=8(range is 5 to 255).

c. Let K3 =400(range is 255 to 999999999).

d. K4= 150 (KEY1 XOR KEY2 XOR (KEY3%256))

e. A List of auto generated Substituent using procedure A

Output: plain text file.

Steps:

```
Step 1.Remove the left most (8 bytes) padding and right most
(8 bytes) padding from the cipher text file .
Step 2 Read each block of plain text of size 16 bytes at a time
in array Temp[] and repeat through step 19 for each such
block.
```

ROUND 1:

```
Step 3 Repeat through step 19 for I =0 to 15.
Step 4 If (I%2) == 0 Then
Step 5 Temp [I] = Temp [I] - KeyStream [I].
        E.g. Temp [I] =72-245= -173.
Step 6 Else Temp [I] =Temp [I] XOR KeyStream [I].
Step 7.Repeat through step 7 while (Temp [I] <0)
Step 8 Temp [I] = Temp [I] + 256.
        E.g. Temp [0] = -173+256=83.
```

ROUND 2:

```
Step 9 CipherByte = Temp [I].
E.g. Let the Temp [0] is 83.
```

Step 10.Repeat through Step 14 for J=0 to 255

Step 11.If (CipherByte =Substituent [J]) then do the following

i). FinalIndex = J .E.g. here J must be 99 according to test example therefore FinalIndex =245.

ii). PreFinalindx2= (FinalIndex XOR K4).

In one second = 1000000 possible key trials

In one hour = 36 x 10^8 possible key trials

In one day = 864 x 10^8 possible key trials

In one year = 3.1536 x 10 ^13 possible key trials (less than half of the total key set).

E.g. PreFinlindex2= (99 XOR 150) = 245.

iii). PreFinalIndex = PreFinlindex2 - K3.

E.g. PreFinalIndex =245-400= -155

iv) Break the loop

Step 12.Repeat through step 13 while (Prefinalindx<0)

Step 13 PreFinalIndex = PreFinalIndex + 256.

E.g. PreFinalIndex = -155+256=101.

Step 14.Prefinlindex= Prefinlindex2-128

E.g. PreFinalIndex = 101-128= -27.

Step 15.Repeat through step 17 for K=0 to 255

Step 16 If (PreFinalIndex =Substituent [K]) then index=K

E.g. Let the index of -27 be 45

i). Evaluate r =index/16 e.g. r=45/16=2. And q=index%16 e.g q=45%16=13.

ii). Evaluate PreOriginalbyte = q*16+r,

E. g PreOriginalbyte = 13*16+2=210

iii) Originalbyte=Originalbyte-128,

E.g. Originalbyte=210-128=82.

iv) Break out of loop.

Step 17 End If

Step 18 Temp [I] = Originalbyte.

Step 19. Write Temp [] array to output file (plain text).

Step 20 Stop.

2.3 Key strength analysis

2.3.1 Analysis for Brute-force attack

1. Key 1 ranges from 0 to 255 i.e. require 8 bits to represent.

2. Key 3 ranges from 255 to 999999999 i.e. it requires 30 bits for representation.

3. Key 4(8 bits).

4. Key 2(8 bits) for left and right padding of the encrypted text to apply diffusion.

5. Key Stream consists of 128 bits (16*8=128). So in total the key size is 182 bit i.e. is there are

2^182 = 6.1299821634635554334334333881086012e + 54 posible keys

Now assume that a hacker have a very fast computer using which he/she can execute our decryption algorithm in 1 micro second for all possible key trials. Even if he tries half the set of keys then also he is quite successful in decrypting.

But then also the hackers require more than one year decrypting the cipher text which is show as below:

3. PERFORMANCE EVALUATION

The proposed procedures are implemented in .net(c#) platform for its available in-built cryptography functionalities. The procedures are implemented successfully for the specified sized input plain texts. Initially, there were constraints for large files such as images or videos where the required primary memory of the system could create a problem in the execution and conversion of the plain text into cipher text. But, later on that problem are also resolved by simply dividing the large file into fixed sized sub-files and then performing the swapping while encoding and decoding. The following table represents the data sets that are obtained during the testing and analysis of the proposed procedure. For the analysis purpose tested results are considered using (k1 = 14, k2 = 8, k3 = 700, k4 = ((k1 ^ k2) ^ (k3 % 256))) the following Dataset:

Table 1 Resultant dataset

File size(in KB)	Encryption in micro second	Decryption in micro second	Increase Length(in KB)
1.07	39.8	2216.9	1.08
2.14	82.9	3901.6	2.15
3.22	163.8	5588	3.23
4.29	211.8	7265	4.30
5.37	243.9	9064.3	5.38
6.44	291.5	10573.3	6.45
7.81	335.8	12863.3	7.82

3.1 Length Analysis

The following graph indicates that the size of the cipher text slightly differs from the plain text size. It also shows that a large file can also be encrypted using the proposed procedure. This is one of our enhancements over our previous work [10] i.e. to keep the output cipher text file size to a limit keeping in mind the bandwidth requirements of the network for large file size especially for ad hoc networks.

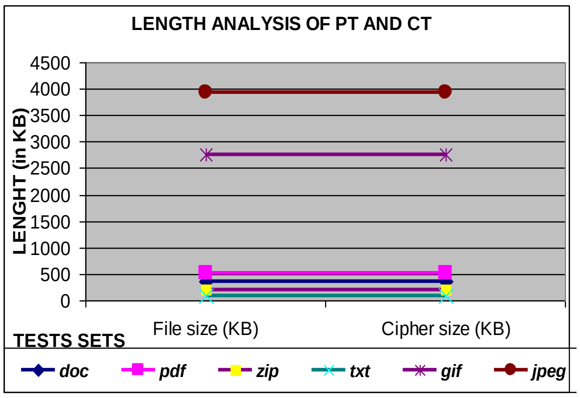

Fig 1: Length analysis of PT and CT

3.2 Time Analysis

The following graph represents the comparisons between the encryption time and decryption time. This graph shows that the decryption time is comparatively much more than that of encryption time. It can also be inferred that a hacker will need more time in brute force attack because the decryption time is more. This is one of the desirable features of cryptography (i.e. to increase the decryption time more than the encryption time so that a hacker needs more time)

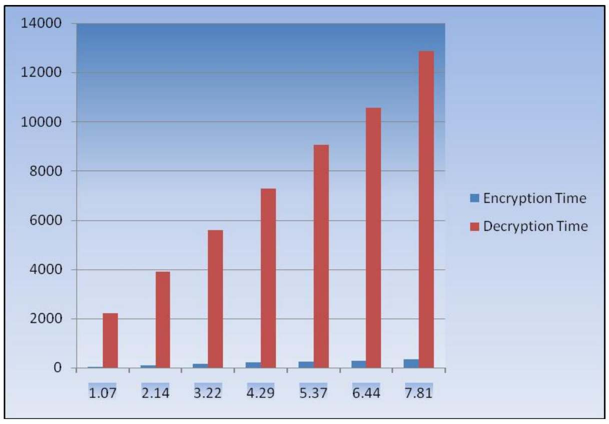

Fig 2: Encrypt time vs. Decrypt time based on file size

4. CONCLUSION

The proposed Encryption/Decryption procedures are the enhanced proposal of previous research work [10]. The procedures are developed keeping in mind the ad hoc networks limited bandwidth. It produce the cipher almost equal to the size of the plaintext which is the main enhancement over [10] where the cipher file size was twice that of its plain text file size. Another important feature in the proposed procedures is that the decryption is more than that of the encryption time. The key stream generation procedure is introduced to generate random set of key streams (using K2 and K3) and making it harder for different attacks. The key strength is increased to four levels of keys. The key3 is the strongest one. The proposal can surely be enhanced with much more advanced concepts such as realization in several security technologies of mobile ad hoc networks where limited bandwidth constraint can be overcome as in our proposed procedure the plain text and cipher text are of same size.

There are a lot of opportunities in expanding and manipulating these ideas towards the cryptographic directions and operations to solve real application especially industrial engineering and management engineering problems. Although this method is efficient, and it is powerful against certain attacks; the two different rounds of algorithms aided with modulus operations and the use of random key stream makes it stronger and faster in execution. Small amount of diffusion is added to make it stronger, but a percentage of increase of cipher size will be there. The list of Substituent is also randomly generated based on the key1.

5 REFERENCES

[1] AtulKahate: “Cryptography and Network security” Tata McGraw Hill Education Pvt. Ltd(2nd edition 2003). Chapter and page no .

[2] William Stallings. *Cryptography and Network Security, Third Edition, Prentice Hall International, 2003.*

[3] Behrouz A.Forouzan:” Cryptography and Network security” McGraw Hill companies(special indian edition, 2007).

[4] Gehani Ashish, La Bean, Thomas H. Reif, JohnH, “DNA-Based Cryptography”, Department of Computer Science, Duke University, June 1999

[5] Guangzhao Cui Limin Qin Yanfeng Wang Xuncai Zhang. An encryption scheme using DNA technology. *Bio-Inspired Computing: Theories and Applications,2008. BICTA 2008. 3rd International Conference on Publication Date: Sept. 28 2008-Oct. 1 2008 ISBN: 978-1-4244-2724-6, page(s):37-42;Adelaide, SA.*

[6] Guangzhao Cui Limin Qin Yanfeng Wang Xuncai Zhang; Information Security Technology Based on DNA Computing; *Anti-counterfeiting, Security, Identification, 2007 IEEE International Workshop on 16-18 April 2007,page(s): 288-291, ISBN: 1-4244-1035-5, Location: Xiamen, Fujian.*

[7] S. V. Kartalopoulos, DNA-inspired cryptographic method in optical communications, authentication and data mimicking, *Proc. of the IEEE on Military Communications Conference, vol.2, pp.774-779, 2005.*

[8] G. Cui, L. Qin, Y. Wang and X. Zhang, Information security technology based on DNA computing, *Proc. of the 2007 IEEE International Workshop on Anti-counterfeiting, Security, Identification, Xiamen, China, pp.288-291, 2007.*

[9] D. Boneh, C. Dunworth and R. Lipton, Breaking DES using a molecular computer, *Technical Report, CS-TR-489-95, Princeton University, 1995*.

[10] Bibhash Roy, Gautam Rakshit, Pratim Singha, Atanu Majumder, Debabrata Datta, “An improved Symmetric key cryptography with DNA Based strong cipher”- ICDeCom-2011, Feb’ 24-25’2011, pp.1-5,2011 Birla Institute of Technology, Meshra, Ranchi, Jharkhand, India pulished at *IEEE Explorer.ISBN:978-1-4244-9189-6, ACCESSION NO: 11887756.*