Encryption of Message Block using Binary Tree in Block Cipher System: An Approach

Sumit Sharma
M.Tech (IS) Student
Department of Computer Science Engineering
AIACTR

Mrs. Shobha bhatt
Assistant Professor
Department of Computer Science Engineering
AIACTR

Abstract

Block cipher system is generally used to encrypt a block of message instead of character by character encryption. Block cipher produces more complexity than stream cipher system. The complexity of block cipher can be increases drastically when we use the binary tree concept for encryption of the message. In this paper, we provide an approach which increases the complexity of the block cipher system and cannot be broken easily. We are also looking the performance of this approach against some standard approaches.

Keywords: Cryptography, Encryption, Decryption, Cipher system, Stream cipher, Block cipher

I. INTRODUCTION

Cryptography deals with two types of operations, one is encryption and other is decryption. For performing encryption or decryption—a series of well-defined steps called procedure is required. In encryption, a message is converted into the secret code which is not easy to understand. Ciphers generally substitute the same number of characters as are input. However, there are exceptions that the number of characters in input as well as in output can be different. In cipher system, original information is known as plaintext, and the encrypted form of plaintext is known as ciphertext. The ciphertext contains all information of the plaintext message, but is not in a readable format. The operation in the cipher system usually depends on a piece of auxiliary information, called a key. The encrypting operation depends on the key. A key must be selected before using a cipher to encrypt a message. Without knowledge of the key, it is impossible to decrypt the resulting ciphertext into plaintext. Modern ciphers can be categorized in several ways: 1) By whether they work on blocks of symbols usually called block cipher or on a continuous stream of symbols called stream cipher. 2) By whether the same key is used for both encryption and decryption called symmetric key cipher system or if a different key is used for encryption and decryption is called asymmetric key cipher system.

Generally cipher system is distinguished by the type of input data, block cipher which used a fixed size of block of input message for encryption and stream cipher which used a stream of characters of the message.

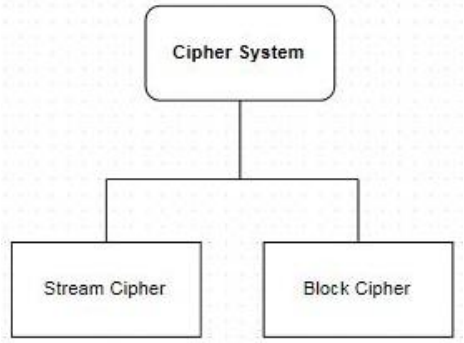

Fig. 1: Diagram

In a stream cipher each character in plaintext is encrypted one at a time with the corresponding character of the key-stream, to give a character of the ciphertext stream. Because of each character is depend on the current state or character so it is also called state cipher. Stream cipher divides in two parts on the basis of the relation of the plaintext and cipertext: 1) synchronous stream cipher where feedback does not required for generating next ciphertext character. 2) self-synchronous stream cipher where feedback required for generating next ciphertext character.

In a block cipher a group of character in plaintext is encrypted at a time with the corresponding key which produces a group of ciphertext characters. Block ciphers are the main components in the cryptography and are widely used to implement on encryption of bulk data. The modern block ciphers are based on the concept of an *iterated* product cipher. Product ciphers were suggested by Claude Shannon in 1949 in his seminal publication whose aim was to improve security by combining simple operations such as substitutions and permutations. Iterated product ciphers carry out encryption in multiple rounds, each of

which uses a different sub-key derived from the original key. An application of the product ciphers is called a Feistel network, which is implemented in the DES. Block ciphers also realized as the AES.

In a publication of the National Institute of Standards and Technology (NIST) in 1977 on DES cipher was the fundamental of modern block cipher design. On the other hand, cryptanalytic attacks also developed parallely. Both differential and linear cryptanalysis arose out by DES design. Today, there are a number of attack techniques against which a block cipher is secure although, brute force attacks are possible on the block cipher systems. Even a secure block cipher is suitable only for the encryption of a single block under a fixed key. A number of ways are designed to repeatedly use of block cipher to provide more security and to achieve the security goals of confidentiality and authenticity. Block ciphers also be used as building blocks in other cryptographic techniques, such as universal hash functions and pseudo-random number generators.

If we implement the binary tree in block cipher encryption then the complexity and the security of the block cipher increases drastically. In this paper, we study an approach which implemented the block cipher encryption using the binary tree concept. We also look for the complexity of this approach against the some standard algorithms or approaches.

II. BINARY TREE IMPLEMENTATION

If we use the binary tree and traverse it using any approach then the complexity of the block cipher increases drastically. For this we first need to break the message into block of the 8 character each. After that, we assign these 8 characters to the leaf nodes of the binary tree of level 3. Other nodes of the binary tree filled using the functions which generates the characters corresponding to the internal nodes of the binary tree. Then we apply another function for transpose the positions of the characters in binary tree. After that we can apply any tree traversal method for generating the ciphertext. In this way, a strong cipher produces and provides the high security in various applications.

We need to choose some parameters for the functions used in encryption and decryption. We use two functions f(x) and g(x). f(x) is used for the diffusion of the characters in the internal nodes of the binary tree and g(x) is used to transpose the characters of the binary tree in order to produce ciphertext. f(x) can be defined as the

$f(x) = ((k.g^x + c) \bmod p)) \bmod 256$

where

p is the prime number greater than length of plaintext
k is the key of encryption in block cipher
g is random number less than prime number
x is the (n-1)th character of the plaintext
c is the positive number less than 8

for limiting the result of f(x), we reduce the result in available ASCII characters using mod 256. After that we create confusion in the binary tree using the another function g(x) which rearranges the values of the nodes. It can be defined as the:

$g(x) = (g^k+c) \bmod 16$

where

k is the key of encryption in block cipher
g is random number less than prime number
c is the positive number less than 16

Now we have new tree that can be traverse to produces more confusion in the ciphertext. we have 4 methods in which we can traverse the tree and produces more confusion. If we have not sufficient characters to complete the tree then we add 'z' as the parity bit to complete the tree.

Decryption is easier than encryption because for it we just need one function. We can decrypt the message just follow the reverse steps of encryption. We just need to using reverse of g(x) function for finding exact values of the nodes and then just collecting the all leaf nodes we get the actual plaintext.

III. FLOW CHART FOR THE ENCRYPTION & DECRYPTION

Encryption process can be shown easily with the help of flow chart. Initially we need some input for starting the encryption process and after that we need to follow steps for encryption the message.

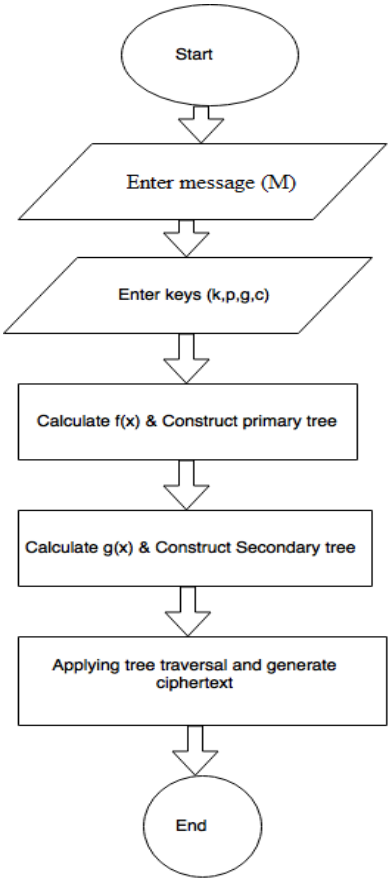

Fig. 2: Flow Chart

Decryption process is easier than the encryption process. It can be shown easily with the help of the flow chart.

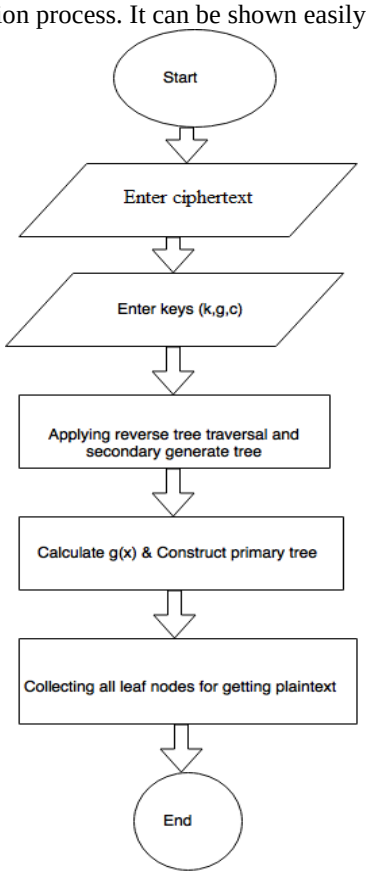

Fig. 3: Flow Chart

A. Complexity calculation for the encryption and decryption:

1) The order of encryption method according to the following algorithm is O($3n^2$), which can be calculated as:

```
Input(message ,key, p, g, c)
V ← 0, z ← 1, y ←  1, flag← 1, l ← 0, c1 ← c, k ← key;
while (l  !=  15) do
for  i←1 to key do
y ←g * y;
form1[l] ← ( y + c) mod 15;
if (l  >=  1) then
for j←0 to l-1 do
if (form1[j] = form1[l]) then
flag←0;
if (flag == 1) then
++l;  ++key,  ++c,
 flag ← 1, y ← 1;
//creates 15 numbers between 0 to 14 with the order of O(2n^2)
For  i←0 to length of (plaintext) – 1 do
for  j←0 to 7 do
bit_text[j] ← j'th bit of plaintext[i];
x ← i;  y← 1;
for j←0 to x-1 do
y ← g * y;
//order of the power according to the last for is O(n^2)
form2 ← ((k * y + c1) mod p) mod 256;
y← 1;
for j<<-0 to 7 do
bit_form2[j] ← j'th bit of form2[i];
for j←1 to 7 do
tree1[j - 1] ← bit_form2[j];
for j←7 to 14 do
tree1[j] ← bit_text[j - 7];
for j←0 to 99 do
tree2[j] ← ' ';
for j←0 to 14 do
tree2[form1[j]] ← tree1[j];
tree4 ← inorder(tree2,tree2[0],0);
//the order of in_order  tour is O(n) but because we know the shape of tree it is O(1)
tree4[15] ← a random bit;
cipher1[2 * i] ← character of tree4[0...7];
cipher1[2 * i + 1] ← character of tree4[8...15];
v ← 0; j ← -1;
for i←0 to length of (cipher1) - 1 do
if (i mod k = 0) then
++j; cipher2[j][i mod k] ← cipher1[i];
//starts the cipher with the order of O(n)
if (length of (cipher1) mod k != 0) then
for u←length of (cipher1) mod k to k-1 do
cipher2[length of (cipher1) / k][u] ← a random character;
//puts the random characters according the key with the order of O(n)
return cipher2;
```

2) The order of decryption method according to the following algorithm is O($2n^2$):

```
Input(cipher, key, g, c)
a ← 0 , k ← key , v ← 0 , m ← 0,l← length of (cipher);
while (t != 15) do
for i← 1 to key do
y ← g * y;
form1[t] ← ( y + c) mod 15;
if (t >= 1) then
for j← 0 to t-1 do
```

```
if (form1[j] = form1[t]) then
flag ← 0;
if (flag = 1) then
++t; ++key,  ++c,  flag ← 1, y ← 1;
//creates 15 numbers between 0 to 14 with the order of O(2n^2)
for i← 0 to l/k-1 do
for j← 0 to k-1 do
cipher[a] ← cipher[(l / k) * j + i];     ++a;
//opens the cipher with the order of O(n)
for j← 0 to 98 do
tree1[j]← ' ';
for j← 0 to 14 do
tree1[j]← 2;invinor ← invinorder(tree1,tree1[0],0);
//finds the positions of a default tree in in_order tour form with the order of O(1)
for i← 0 to l-1 do
for j ← 0 to 7 do
bit_ciph[j] ← j'th bit of ciph[i]
for j← 8 to 15 do
bit_ciph[j] ← (j-8)'th bit of ciph[i+1]
for j← 0 to 14 do
inorder[j] ← bit_ciph[j];
for j← 0 to 14 do
tree1[invinor[j]] ← inorder[j];
for j← 0 to 14 do
tree2[form1[j]]← tree1[j];
for j← 7 to 14 do
bit_text[j-7]← tree2[j];
text[m]← character of bit_text[0…7];
++i,  ++m;
//decrypts the tree cipher with the order of O(n)
return text;
```

IV. STRENGTH OF THIS APPROACH

1) If the attacker wants to break this cipher than the brute force is the only way to break this.
2) Instead of using the limited characters, we use a random character from the 256 characters of ASCII code. This makes the cipher more safer.
3) Because we use two functions to fill the binary tree and in-order tour so we can give a key more than the length of the plaintext.
4) The only way to break this cipher system is the brute force attack and make exhausted searches with the complexity of ((28)2)n in which it represents the double sized cipher text and n is the plaintext length.

V. CONCLUSION

When we use the binary tree for the encryption of the plaintext then it convert the 8 byte into 16 byte i.e. double the size of the plaintext. In all cipher systems length of the ciphertext is equal to the length of the plaintext so, this cipher system increases the complexity and it is hard to attackers to break this cipher system. The plaintext of eight or more characters would provide more resistance to a brute force attack in comparison to 128-bit AES and has the same resistance to brute force attack as 256-bit AES.

This methodology is very suitable in the areas where no problem of storage. It is not suitable for the areas with low bandwidth or limited storage capacity. However for most communication channels where encryption is required, an increase in the plaintext size will not have a significant impact. Also using a good random generator in function F(x) can make the cipher effective.

REFERENCES

[1] Massoud Sokouti, Saeid Pashazadeh,2010, An approach in improving transposition cipher system, IJST
[2] Okike Benjamin,E.G.D. Garba,2013, Pattern In Splitting Sequence In Okike's Merged Irregular Transposition Cipher Technique In Securing Web Informatio,IJES.
[3] Vaudenay S,2006, A Classical introduction to modern cryptography, Springer.
[4] William Stallings,2011,Cryptography and network Security Principles and Practic, 4th edition,pearson.
[5] Atul kahate,2007,Cryptography and Network Security,Tata McGrow Hill.

[6] Behrouz A. Forouzan,2004, Cryptography and Network Security,Tata McGraw Hil.
[7] AL.Jeeva, Dr.V.Palanisamy, K.Kanagaram,2012, comparative analysis of performance efficiency and security measures of some encryption algorithms,IJERA.