Proposed Method to Enhance the Security of Blowfish Algorithm

Dr.Shaimaa H. Shaker*

Received on: 8 /3 / 2011
Accepted on: 8 /9/2011

Abstract

The selective application of technological and related procedural safeguards is an important responsibility of every organization in providing adequate security to its electronic data systems. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data.

This paper introduces a proposed method to enhance the complexity of the Blowfish algorithm. This is done by using an operation depends on using eighteen multiple keys, each key consists of a combination of 4 states (0, 1, 2, 3) instead of the ordinary 2 state key (0, 1) applied during the 16 round of Blowfish algorithm, according the analysis of results found the new approach provide more secure and more robustness to cryptanlaysis methods.

Keywords: computer security, Blowfish algorithm, triple encryption .

طريقة مقترحة لتحسين امنية خوارزمية Blowfish

الخلاصة

اختيار التطبيق للاجراءات التقنية الوقائية ِ وذات العلاقةِ تمثل مسؤوليةُ مهمةُ لكُلّ منظمة في تَزويد الأمنِ الكافي إلى أنظمةِ بياناتِه الإلكترونيةِ. حماية البياناتِ أثناء الإرسالِ أو الخزن قَدْ تَكُون ضروريَ لإبْقاء السريّةِ وسلامةِ المعلوماتِ . هذا البحث يقدم طريقة مقترحة لتعزيز تعقيد خوارزمية Blowfish حيث تعتمد هذه الطريقة على استخدام ثمانية عشر مفتاح متعدد كل مفتاح يتكون من اربع حالات وهي (0,1,2,3) بدلا من استخدام المفتاح الاعتيادي والذي يتكون من حالتي(0,1) والمطبقة خلال 16 دورة في خوارزمية الblowfish وطبقا لتحليل النتائج وجد بان الطريقة الجديدة توفر امنية اكثر وممانعة اكثر لطرق تحليل الشفرة.

1. Introduction

Cryptography has a role to play in many areas. Like seat belts, it will not completely protect us. There has been a symbiotic relationship between cryptography and the development of high-performance computing systems. Cryptography is especially useful in the cases of

Financial and personal Data, irrespective of the fact that the data is being transmitted over a medium or is stored on a storage device. It provides a powerful means of verifying the authenticity of data and identifying the culprit, if the

***Computer Engineering Department, University of Technology/Baghdad**

https://doi.org/10.30684/etj.29.13.6
2412-0758/University of Technology-Iraq, Baghdad, Iraq

confidentiality and integrity of the data is violated. Because of the development of electronic commerce, cryptographic techniques are extremely critical to the development and use of defense information systems and communications networks. Every security system must provide a bundle of security functions that can assure the secrecy of the system. These functions are usually referred to as the goals of the security system. These goals can be listed under the following five main categories: [1]

Authentication:

This means that before sending and receiving data using the system, the receiver and sender identity should be verified.

Secrecy or Confidentiality:

Usually this function (feature) is how most people identify a secure system. It means that only the authenticated people are able to interpret the message (date) content and no one else.

Integrity:

Integrity means that the content of the communicated data is assured to be free from any type of modification between the end points (sender and receiver).

Non-Repudiation:

This function implies that neither the sender nor the receiver can falsely deny that they have sent a certain message.

Service Reliability and Availability:

Since secure systems usually get attacked by intruders, which may affect their availability and type of service to their users.

Such systems should provide a way to grant their users the quality of service they expect. One of the most important protection methods is encryption algorithms. These algorithms uniquely define the mathematical steps required to transform data into a cryptographic cipher and also to transform the cipher back to the original form. This paper introduces a research which is an attempt to improve Blowfish algorithm that depend on using logical XOR operation[2]. To give more prospective about the performance of the algorithms, introduce the Some other resources related with this work, Nadeem2005[3] introduce the popular secret key algorithms including DES, 3DES, AES (Rijndael), Blowfish, were implemented, and their performance was compared by encrypting input files of varying contents and sizes. Hala2009[4] proposed new quantum cryptography system using quantum description techniques. Rehab2010[5] proposed a new approach for modifying DES algorithm using 4-state Multi-keys. In these papers enhance complexity by increase the computations. The improved blowfish in this paper enhance the complexity of blowfish algorithm by using more than one key with multi-state in each round, change the F function and using # operation.

2. Blowfish Algorithm

It is one of the most common public domain encryption algorithms provided by Bruce Schneier -one of the world's leading cryptologists, and the president of Counterpane Systems, a consulting firm specializing in cryptography and computer security. Blowfish is a variable length key, 64-bit block cipher. The Blowfish algorithm was first introduced in 1993.This algorithm can be optimized in hardware applications though it's mostly used in software applications. Though it suffers from weak keys problem, no attack is known to be successful against[2][3].

2.1Description of Blowfish Algorithm

Blowfish is a 64-bit block cipher with a variable-length key. The algorithm consists of two parts: key expansion and data encryption. Key expansion converts a key of up to 448 bits into several sub-key arrays totaling 4168 bytes. Data encryption consists of a simple function iterated 16 times. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are additions and XORs on 32-bit words. The only additional operations are four indexed array data lookups per round, see figure(1).

Blowfish uses a large number of subkeys. These keys must be precomputed before any data encryption or decryption. The P-array consists of eighteen 32-bit subkeys: $P_1, P_2, \ldots, P_{18}$.

Four 32-bit S-boxes have 256 entries each:

$S_{1,0}, S_{1,1}, \ldots, S_{1,255}$

$S_{2,0}, S_{2,1}, \ldots, S_{2,255}$

$S_{3,0}, S_{3,1}, \ldots, S_{3,255}$

$S_{4,0}, S_{4,1}, \ldots, S_{4,255}$

The exact method used to calculate these subkeys as follows:

(1) Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of p.

(2) XOR *P*1 with the first 32 bits of the key, XOR *P*2 with the second 32-bits of the key, and so on for all bits of the key (up to *P*18). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits.

(3) Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).

(4) Replace *P*1 and *P*2 with the output of step (3).

(5) Encrypt the output of step(3)using the Blowfish algorithm with the modified subkeys.

(6) Replace *P*3 and *P*4 with the output of step (5).

(7) Continue the process, replacing all elements of the P-array, and then all four S-boxes in order, with the output of the continuously changing.

Blowfish is a Feistel network consisting of 16 rounds. A Feistel network is a general method of transforming any function (usually called a F function) into a permutation. It was invented by Horst Feistel and has been used in many block cipher designs. The working of a Feistal Network is given below:

§ Split each block into halves

§ Right half becomes new left half

§ New right half is the final result when the left half is XOR'd with the result of applying *f* to the right half and the key.

The input of blowfish algorithm is a 64-bit data element, *x*. To encrypt: Divide *x* into two 32-bit halves (x_L, x_R) where:

For $i = 1$ to 16:

$x_L = x_L \bullet P_i$

$x_R = F(x_L) \bullet x_R$

Swap x_L and x_R

Swap x_L and x_R (Undo the last swap.)

$x_R = x_R \bullet P_{17}$

$x_L = x_L \bullet P_{18}$

Recombine x_L and x_R.

Function F is as follows :

Divide x_L into four eight-bit quarters: *a*, *b*, *c*, and *d*

$F(x_L) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \bullet S_{3,c}) + S_{4,d} \bmod 2^{32}$.

Decryption is exactly the same as encryption, except that $P_1, P_2,..., P_{18}$ are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all sub-keys are stored in cache. In total, 521 iterations are required to generate all required sub-keys. Applications can store the sub-keys there's no need to execute this derivation process multiple times[2][6].See figure(2).

3. The Proposed Algorithm to Enhance Blowfish using Multi-States of Multiple Keys

This research proposed an improvement to the Blowfish algorithm. The proposed improvement makes use of operation (#) applied during each round in the original Blowfish algorithm, to apply operation (#) another key is needed. The output of the four S-boxes with improvement blowfish also applied operation(#) in each round, see figure(5). Section 3.1 introduce the description of multi-states operation , section 3.2 explain the details of the additional key used in operation (#) to improve blowfish algorithm.

3.1 Description of Multi- States operation

This paper used # operation which deals with multiple inputs each of these inputs has multi-states(0,1,2,and3). To increase the security and key space, that makes the encryption algorithms more robustness to the intruders, a new manipulation bits process has been added in [4] by using different truth table for manipulation bits process work on 4-states (0,1,2,and3) , while the traditional binary process (XOR) work on (0, 1) bits only. The symbol # has been used to refer to the operator that execute this process use truth tables that shown in figure (3)[4]. The new operation needs 3 inputs, the first one specify the table number that should be used to calculate the result among the 4 tables, the other 2 inputs define the row and column number in the specified table where the cross point of them gives the result.

An example of applying the #operation is shown below table(1):[5]

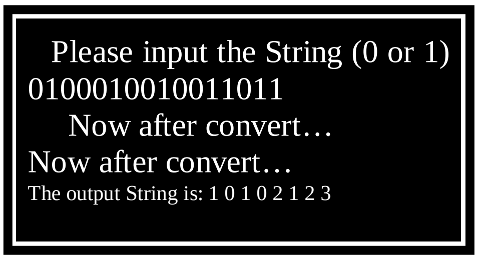
when applying # operation on the three inputs using the truth table in figure(3) , the output is 1 1 1 2 3 2 0 0 3 1 3 0 also multi-state of (0,1,2,3). The current paper using this operation to enhance the complexity of blowfish algorithm figure(5) and figure(6).

3.2 Using Multi-State of Multiple Keys

Additional key used in improvement blowfish may come in binary form and convert to a 4-states key, or it may already come in a 4-states as that can be done with quantum channel. Multiple keys will be used in each round of the original Blowfish , the first key K_i will be used with the F function, K_i consists of the hexadecimal digits convert to binary form. The second key will be the first input to the # operation, the second input will be the output of the **F** function, and the third input to the # operation will be the value R_i, Figure (4) shows the three 32-bits input to the # operation , and the 32-bits output, with the places needed to convert these 32- bits to 16-digits,figure(5) shows the details of each single improvement

blowfish round, figure(6) shows the structure of encryption operation of improvement blowfish algorithm.

The improvement blowfish encryption like the origin blowfish consists of 16 round with 18 Ps'. These three inputs to the # operation should be firstly converted from 32 bits to a 16 digits each may be one of four states (0, 1, 2, 3), i.e., each two bits converted to its equivalent decimal digits, using convertor function see algorithm(1),for example, the binary number:



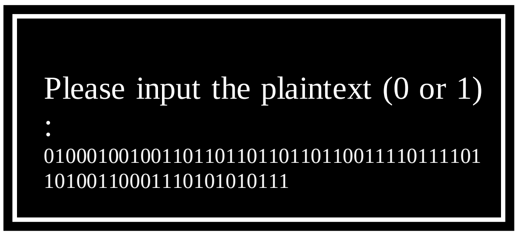
```
  Please input the String (0 or 1)
0100010010011011
    Now after convert…
Now after convert…
The output String is: 1 0 1 0 2 1 2 3
```

Please input the String (0 or 1) :
The output String is: 1 0 1 0 2 1 2 3

Where R_{i-1} represents
Then the # operation will be applied to generate a new 16 digits that should be reconverted to 32 bits ,see algorithm(2) to be the input to Li+1, see Figure (4). Full details of the proposed improved Blowfish are given in Algorithm (3).

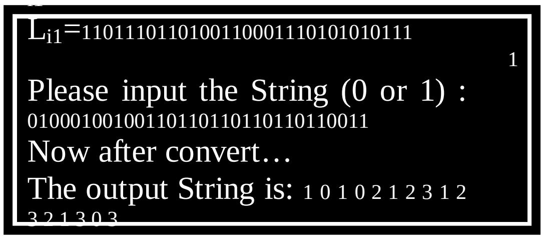
4. Implementation of Proposed Algorithm

This section explain the encryption and decryption operations with example of improvement blowfish results according to apply the # operation. First enter the plaintext ,divided it into 32bit right part and 32bit left part converted to 16 digit(0,1,2,3).

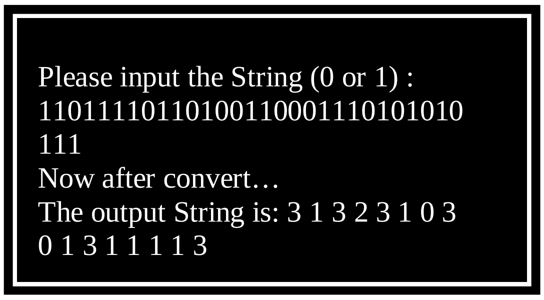


```
Please input the plaintext (0 or 1)
:
0100010010011011011011011011001111011110 1
1010011000111010101010111
```

Then R_{i-1} and L_{i-1} after convert will



```
Li1=11011110110100110001110101010111
                                                1
Please input the String (0 or 1) :
01000100100110110110110110110011
Now after convert…
The output String is: 1 0 1 0 2 1 2 3 1 2
3 2 1 3 0 3
```



```
Please input the String (0 or 1) :
110111101101001100011101010
111
Now after convert…
The output String is: 3 1 3 2 3 1 0 3
0 1 3 1 1 1 1 3
```

The output String is: 1 0 1 0 2 1 2 3
1 2 3 2 1 3 0 3
Please input the String (0 or 1) :
Now after convert…
The output String is: 3 1
110111101101001100011101010
1113 2 3 1 0 3 0 1 3 1 1 1 1 3

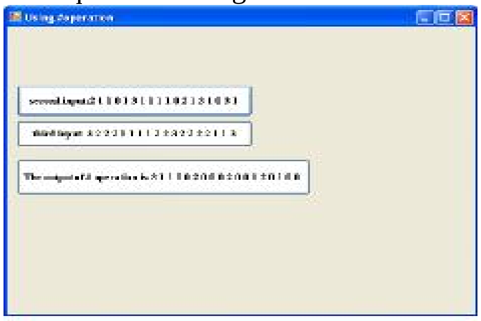
the (1st input to operation #).
R_{i-1}=01000100100110110
1101101101 10011
R_{i-1}'= 1 0 1 0 2 1 2 3 1 2 3 2 1 3 0 3

Applying the function **F**, which represents here the first key(2nd input to operation #). The proposed improvement blowfish used 3 input to operation (#) with S-boxes part, the 16 digit of (0,1,2,3) as the result of (+)operation between the output of 32-bits of s-box1 and 32-bits of s-box2,and this represent the 1st input to operation #. The 32-bit of s-box3 convert to 16 digit of (0,1,2,3) represent the2nd input to operation #.

The 32-bit of s-box4 convert to 16 digit of (0,1,2,3) represent the3rd input to operation #. The output of operation # is 16 digit of (0,1,2,3) as **F**,see algorithm(2). **F=2 1 1 0 1 3 1 1 1 1 0 2 1 3 1 0 3 1** , Now enter the key, which represent the second key(3rd input to operation #), **P_i=11101010010101011 0101110101010010111**,converted to 16 digit(0,1,2 ,3) then **P_i'= 3 2 2 2 1 1 1 1 2 2 3 2 2 2 2 1 1 3.** Then the # operation applied according to truth tables shown in figure(3) , the result of improvement blowfish encryption will be:

Li'= 3 1 1 0 0 2 0 0 0 2 0 0 1 2 0 1 0 0

For decryption operation one can reverse the whole operation then will get the initial number, which is the result of the encryption operation that equal to the original data:



second input:2 1 1 0 1 3 1 1 1 1 0 2 1 3 1 0 3 1

third input: 3 2 2 2 1 1 1 1 2 2 3 2 2 2 2 1 1 3

The output of # operation is:3 1 1 0 0 2 0 0 0 2 0 0 1 2 0 1 0 0

P_i'= 3 2 2 2 1 1 1 1 2 2 3 2 2 2 2 1 1 3
F= 2 1 1 0 1 3 1 1 1 1 0 2 1 3 1 0 3 1
L_i'= 3 1 1 0 0 2 0 0 0 2 0 0 1 2 0 1 0 0

R_{i-1}'= 1 0 1 0 2 1 2 3 1 2 3 2 1 3 0 3
R_{i-1}=01000100100110110110110110 11 01100 11

5.Discussion of the Results

The security of improved Blowfish algorithm will be increased by using addition key while the block size and key length were still the same, using # operation applied during each round that can be done with quantum channel. The number of iterations which will be required to break the two keys will be increased.

The block size of 64-bits makes Blowfish algorithm vulnerable to the matching ciphertext attack. Where after encryption of 2^{32} blocks, equal ciphertexts can be expected and information is lacked about plaintext. So that, the improved Blowfish algorithm is resistant to matching ciphertext attack because it used additional key . Horst Feistel[7] referred to the avalanche effect as: “a small change in the key gives rise to a large change in the ciphertext”. Based on that using two different keys gives rise to a large change in the ciphertext. Improved algorithm is secure, compact and simple block cipher. It adopts key-dependent permutations and substitutions to provide protection against differential cryptanalysis and linear cryptanalysis. Each function of the proposed algorithm is dependent on its key, this wills prevent fixed output and increase the non-linearity of the algorithm. Because of the large key and input block size of the proposed algorithm, exhaustive key search and the matching ciphertext attack are infeasible. The improvement blowfish algorithm proposed to modify F by replacing the XORoperation by # operation so it could reduces the number of operation used in F and still the complexity of the improvement blowfish high,also the time that require to do all operations is equal to that time used with blowfish algorithm see table(2)figure(7).

6. Conclusions

Although key dependent S-boxes and sub-keys, generated using cipher itself, makes analysis very difficult and changing both halves in each round increases security, still need more level of security. Because of brute-force key search is practical when provided key is not large enough, additional key is used to reduce the probabilities of break key. So it becomes very important to augment this improvement blowfish algorithm by adding new levels of security to make it applicable and can be depending on in any common communication channel ,so multiple state adding more probability to guess the origin key. Adding additional key and replacing the old XOR by a new operation as proposed by this paper to give more robustness to Blowfish algorithm and make it stronger against any kind of intruding. The ciphering process stills simple and can be implemented by hardware in this new proposed improvement, as well as the time complexity of the new algorithm stays the same since only one operation is replaced by another operation, and the conversion operations is very simple and straightforward.

7. References

[1] Aron E. Earle, "Wireless Security handbook", Aerbach publication,2005.

[2] William Stanlings, "Cryptography and Network Security: Principles and Practice (5th Edition)", Tata McGraw Hill, 2010

[3] Aamer Nadeem el al, "A performance comparsion of data encryption algorithm", IEEE [information and communication technologies], 2005.

[4] Hala Bahjat AbdulWahab, Abdul Monem S. Rahma, 'Proposed New Quantum Cryptography System Using Quantum Description techniques for Generated Curves", The 2009 International conference on security and management, SAM2009, July 13-16 2009, Las Vegas, USA, SAM 2009.

[5] Rehab F.Hassan,"New approache for modifying DES algorithm using 4-state Multi-keys",Eng. And Tech.journal, Vol.28,No20,2010.

[6] Henk C.A. van Tilborg, Eindhoven , "ENCYCLOPEDIA of CRYPTOGRAPHY and SECURITY ", Springer Science+Business Media, Inc, 2005.

[7]Alan G. Konheim,"COMPUTER SECURITY and CRYPTOGRAPHY ",by John Wiley & Sons, Inc. 2007 .

Table (1) an example of applying the #operation

1st input	1	1	0	3	2	2	1	1	0	3	1	2
2nd input	2	3	0	1	3	1	1	1	0	0	2	3
3rd input	3	2	2	2	2	1	1	0	0	1	1	1
Output	1	1	1	2	3	2	0	0	3	1	3	0

Table(2): performance analysis

block size(bits)	blowfish (kilobits/sec)	improv_blowfish(kilobits/sec)
64	6019	6116
128	27774	27552
256	53783	53888
512	109599	109976

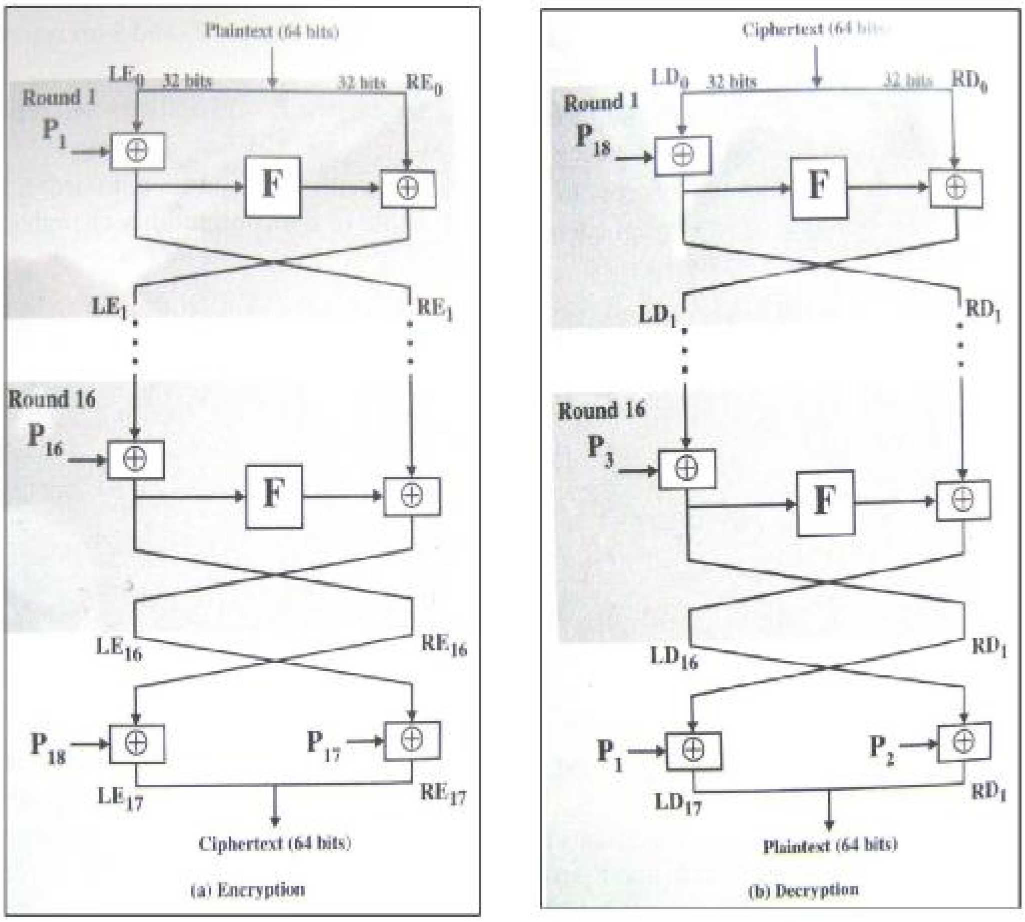


Figure (1) Blowfish Encryption and Decryption

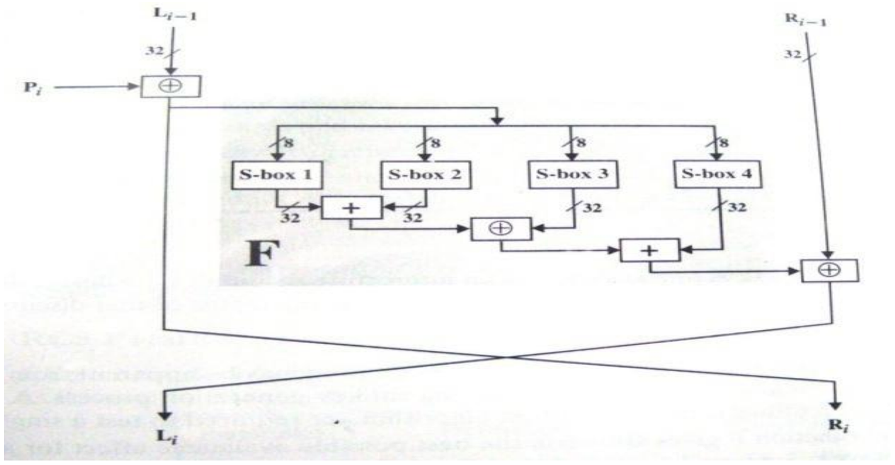


Figure (2) Blowfish Single Round

#0	0	1	2	3
0	3	2	1	0
1	2	3	0	1
2	1	0	3	2
3	0	1	2	3

#1	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

#2	0	1	2	3
0	2	3	0	1
1	3	2	1	0
2	0	1	2	3
3	1	0	3	2

#3	0	1	2	3
0	1	0	3	2
1	0	1	2	3
2	3	2	1	0
3	2	3	0	1

Figure (3): Truth Table of # operation

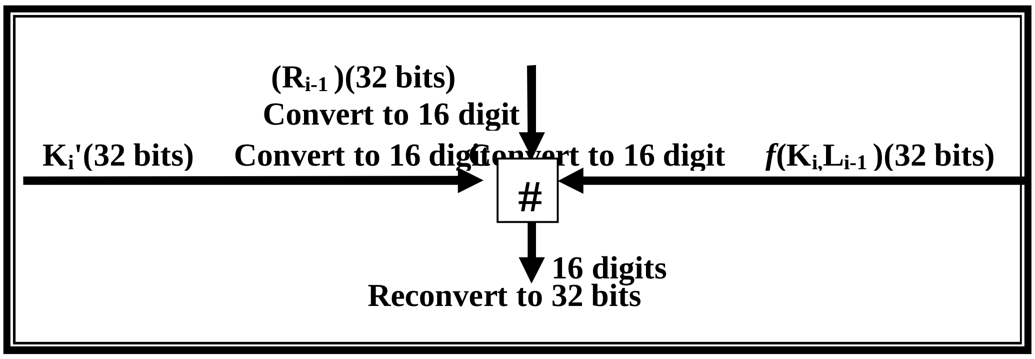


Figure (4) Inputs and Output of the # operation in Blowfish algorithm

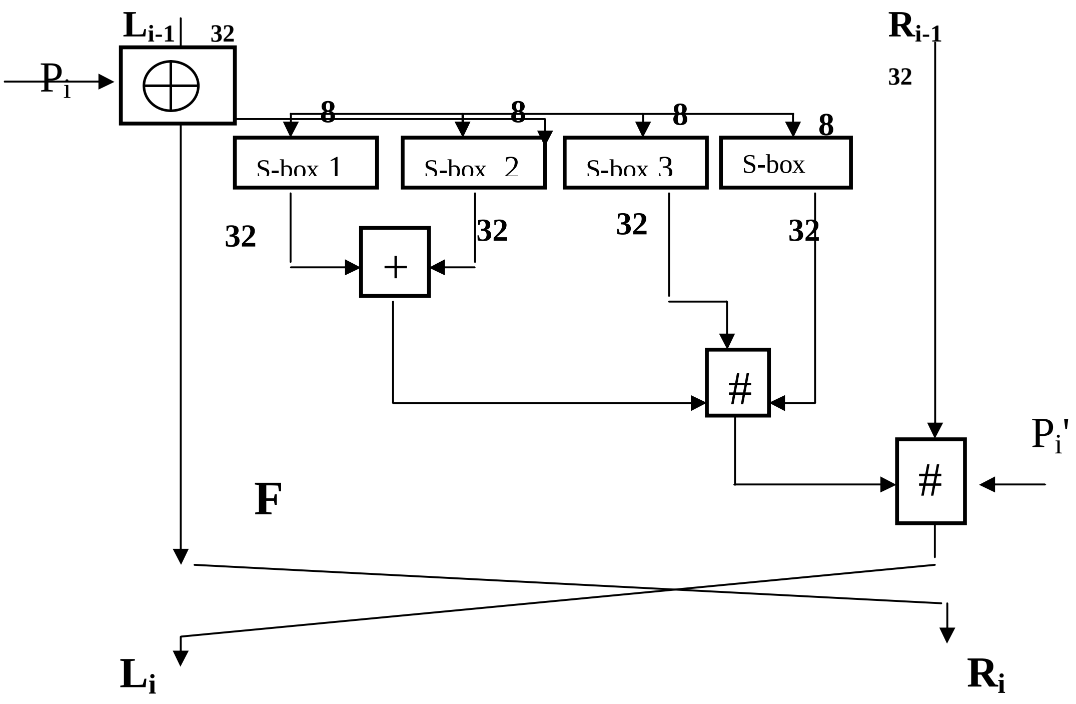


Figure (5) Detail of single improvement Blowfish Round.

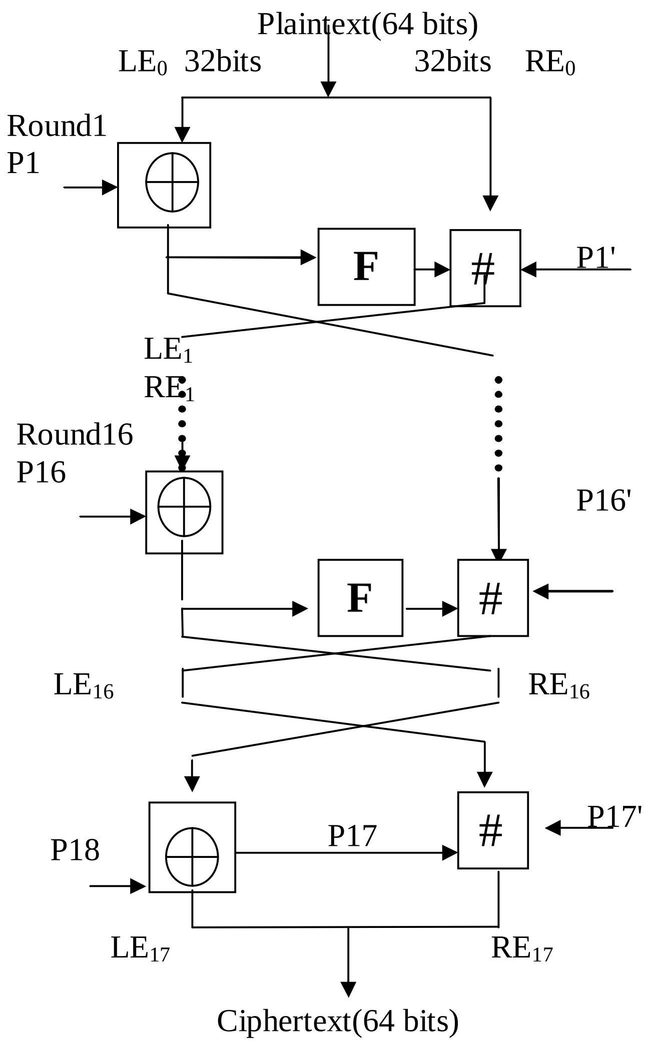


Figure (6) Improvement Blowfish Encryption.

```
Algorithm(1)

Converted1 algorithm;
Input:32 bit of(0,1,2,3)
Output:16 digit of (0,1,2,3)
Begin
    1- Divided the 32bit into Blocks of 2bit.
    2- Convert each Block to binary numbers
       -if 0 0 then 0
       -if 0 1 then 1
       -if 1 0 then 2
       -if 1 1 then 3
    3- Output the result as 16 digit of (0,1,2,3);
    end
```

```
 Algorithm(2)
Improvement s-boxes algorithm;
Input:32-bit of (0,1);
Output:16-digit of (0,1,2,3);
Begin
    1- Divided 32-bit into 4 Blocks, each  Block-length-bit;
    2- Enter each 8-bit to s-box, have 4-boxes;
    3- Output 32-bit of (0,1) for each s-box(using the substitution
       method);
    4- Apply + operation
           • The first input is the 32-bit from s-box1
           • The second input is the 32-bit from s-box2
           • The output of the + operation is  32-bit
  Note:  + is addition modulo $2^{32}$
    5- Apply # operation
           • The first input is the 32-bit of step 4
           • The second input is the 32-bit from s-box3
           • The third input is the 32-bit from s-box4

   End
```

```
Algorithm(3)
Encryption of improvement Blowfish algorithm;
Input: 64 bits data of plain text
Output:64 bits data of ciphertext
Begin
1-compute eighteen 32-bit subkeys K, from K.
2-compute 18 subkey of 32-bit $K_i$',from K'.
3- data is divided into two 32-bit halves $L_0$ & $R_0$
2-for $i$ = 1 to 16 do
    Convert $L_i$&$R_i$ into {0,1,2,3}
    $R_i = L_{i-1}$ XOR $P_i$;
   $L_i$ = F[$R_i$] # $R_{i-1}$# $P'_i$;
   $L_{17} = R_{16}$ #$P_{18}$;
  $R_{17} = L_{16}$ # $i_{17\#}$ $P'_{18}$;
3- reconvert $L_i$&$R_i$ into (0,1)
4- the operation # in $L_i$ is computed as follows:
    • Convert the 32-bits resulted from $f(L_{i-1},K_i)$ into 4-state 16 digit call
      F .
    • Convert the 32-bits of $R_{i-1}$ to 4-state 16 digits call it $R_{i-1}$.
    • Convert the 32-bits of Ki' to 4-state 16 digit.
    • Compute $L_i$ by applying the # operation on f according figure(3)
Where F[$a,b,c,d$] = #(($S_{1,a}$ + $S_{2,b}$) , $S_{3,c}$, $S_{4,d)}$
Note:   + is addition modulo $2^{32}$
End.
```

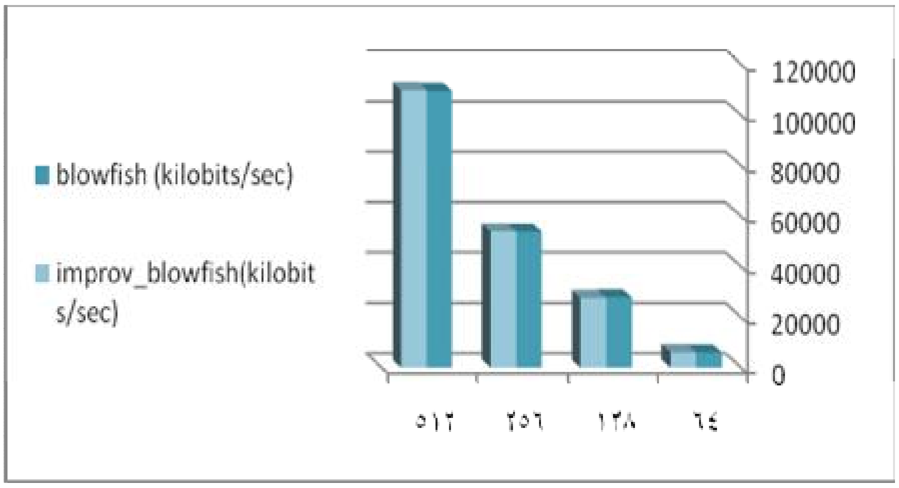


Figure (7): performance analysis of improvement blowfish algorithm