Design and Implementation of an Improvement Of Blowfish Encryption Algorithm

Ashwaq T. Hashim* Dr. Saleh M. Al-Qarrawy* & Janan A. Mahdi*

Received on: 23/4/2008
Accepted on: 9/7/2009

Abstract

Block cipher is a major part of cipher algorithm like stream cipher and other techniques. Its power comes from dealing with plaintext as parts and operating on each block independently. Blowfish is a secret-key block cipher proposed by B. Shneier. It is a Feistel network, iterating a simple encryption function 16 times. The block size is 8-bytes and the key can be any length up to 56-bytes.

In this paper, a Blowfish was improved to encrypt 16-bytes using a variable key length from 8-bytes up to 144-bytes. During the design of *Improved Blowfish algorithm*, the pragmatic aim was to satisfy as many goals as possible while keeping the cipher simple. Only by keeping a cipher simple one can achieve a well-understood level of security, good performance, and a versatility of design that makes the cipher highly adaptable to future demands. The improved algorithm reduced the memory requirement by using a single S-box instead of four S-boxes without compromising security.

The security of improved Blowfish algorithm will be increased by several techniques where the block size and key length were increased, using more complex function before the first round and after the last round and using a complex function to avoid a symmetric to the output of S-box.

الخلاصة

التشفير الكتلي جزء رئيسي من خوارزمية التشفير مثل التشفير الانسيابي وبقية التقنيات الأخرى. قوّته جاءت من التعامل مع plaintext كأجزاء تتعامل مع كلّ كتلة بشكل مستقل. Blowfish هو تشفير كتلي ذو مفتاح سري إقترح من قبلB. Shneier . وهو شبكة فيستيل، يكرر وظيفة تشفير بسيطة 16 مرة. إنّ حجم الكتلة 8-bytes والمفتاح يمكن أن يكون أيّ طول إلى 56-bytes.

تم تحسين الBlowfish في هذا البحث لتشفيرbytes -16 مستعملا مفتاح متغيّرالطول من 8-bytes إلى 144-bytes. أثناء تصميم خوارزمية Blowfish المحسّنة، الهدف الواقعي كان لتحقيق الاهداف قدر ما هو محتمل بينما يبقي التشفير بسيط. فقط بواسطة الإستمرار بالتشفير البسيط الواحد يمكن أن ينجز مستوى مفهوم بشكل جيد من الأمن , أداء جيد، و تعددية الاستعمال للتصميم التي تجعل التشفير متكيّفجدا إلى الطلبات المستقبلية. خفّضت الخوارزمية المحسّنة متطلب الذاكرة بإستعمال صندوق واحدا بدلا من أربعة صناديق بدون مساومة الأمنية. أمن خوارزمية Blowfish المحسّنة سيكون متزايد بعدّة تقنيات حيث تم زيادة حجم الكتلة والطول الرئيسي وإستعمال وظيفة أكثر تعقيدا قبل الدورة الأولى وبعد الدورة الاخيرة واستخدام وظيفة معقّدة لتفادي التماثل إلى ناتج الصندوق(S-box).

*

1. Introduction

The security of symmetric cryptosystem is a function of two parameters: the strength of the algorithm and the length of the key. The algorithm must be so secure that there is no better way to break it than with a brute-force attack. The security of the algorithm must reside in the key, therefore, there is a balance between choosing long key and the time required to complete the enciphering operation [1].

The name of block cipher came from the fact that block cipher encrypts plaintext as blocks. These blocks differ in size between block cipher algorithms, for example, in DES the plaintext is divided into blocks of length 64, but it is 32 in IDEA. If the length of block cipher equal one then, it will become stream cipher. If the length is large then some attacks will work better.

The basic ingredients of modern fast software block encryption schemes are computer instructions like ROTATE, ADD, XOR etc. Different subsets of such operations will yield an interesting variety of different permutation groups, e.g. symmetric groups. For example simple pair of ROTATE and an ADDITION module is already powerful enough to generate every possible encryption function on its set of input blocks. On the other hand, any possible combination of ROTATE and XOR operations can only produce a subset of at most $n \times 2^n$ functions within the symmetric group of order n ! [2].

The fixed initial and final permutations of DES have been long regarded as cryptographically worthless. Khufu XORs the text block with key material at the beginning and the end of the algorithm.

This paper present some existing modern cipher such as ABC, a substitution-permutation network comprising 17 rounds with 3 different kinds of round functions. It is derived from MMB and SAFFER block cipher [3] and Unbalanced Feistel Networks and Block-Cipher Design (UFNs) consist of a series of rounds in which one part of the block operates on the rest of the block [4]. And also PRESENT: An Ultra-Lightweight which is block cipher. It is an example of an SP-network and consists of 31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported [5].

2 Blowfish Algorithm

Blowfish is a block cipher that encrypts data in 8-byte blocks. The algorithm consists of two parts: a key-expansion part and a data-encryption part. Key expansion converts a variable-length key of at most 64 bytes (512 bits) into several subkey arrays totaling 4168 bytes [6].

2.1 Subkeys

Blowfish uses a large number of subkeys. These keys must be precomputed before any data encryption or decryption [6].

The P-array consists of 18 32-bit subkeys:

$P_1, P_2, \ldots P_{18}$.

There are also four 32-bit S-boxes with 256 entries each:

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

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

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

*

$S_{4,0}$, $S_{4,1}$,…., $S_{4,255}$;

2.2Encryptionand Decryption

The underlying philosophy behind Blowfish is that simplicity of designed yields algorithm that is easier to implement. Through the use of a streamlined Feistel network and a simple S-box substitution and a simple P-box substitution. Feistel network makes up the body of the blowfish is designed to be as simple as possible, while still retaining the desirable cryptographic properties of the structure.

Figure (1) illustrates the architecture of the Blowfish algorithm with 16-rounds. The input is a 64-bit data element, X, which is divided into two 32-bit halves: XL and XR

For I= 1 to 16:
XL = XL XOR Pi
XR = F (XL) XOR XR
swap XL and XR

After the sixteenth round, swap XL and XR again to undo the last swap. Then,
XR = XR XOR P17 and
XL = XL XOR P18.

Finally, recombine xL and xR to get the ciphertext [7].

Each bit of the xL is only used as the input to one S-box. In DES many bits are used as inputs to two S-boxes, which strengthen the algorithm considerably against differential attacks.

The number of rounds is 16 and this number affects the size of the P-array and therefore the subkey-generation process; 16 iterations permits key lengths up to 512 bits.

2.3 Function F:

The function F is as follows, see Fig. 2, [7]:

Divide XL into four eight-bit quarters: a, b, c, and d. Then,

$F(xL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}$.

The non-reversible function is designed for strength, speed, and simplicity. The function that combines the four S-box outputs is as fast as possible. A simpler function would be to XOR the four values, but mixing addition mod 2^{32} and XOR combines two different algebraic groups with no additional instructions. The alternation of addition and XOR ends with an addition operation because an XOR combines the final result with XR.

Decryption process is exactly the same as encryption, except that P1, P2,…….., P18 are used in the reverse order.

2.4 Subkeys Generation

The subkeys are calculated using the Blowfish algorithm 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 pi (less the initial 3):
   P1 = 0x243f6a88,
   P3 = 0x13198a2e,
   P2 = 0x85a308d3,
   P4 = 0x03707344, etc.
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P16). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)

*

3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2)
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times [7].

3. The improved Algorithm

In general the improved algorithm differs from the previous Blowfish algorithm, which encrypts and decrypts 64-bit block size and the key can be any length up to 512 bits. The improvement algorithm is used to encrypt and decrypt 128 bits block size and the key can be any length up to 1152 bits (144 bytes). It is aimed at decreasing memory requirement by fewer and smaller S-boxes. Figure (3) shows the structure of improvement algorithm. It consists of splitting the plaintext into two 64-bits halves. Feistel ciphers are a special class of iterated block ciphers, where the ciphertext is calculated from the plaintext by repeated application of the same transformation or round function. In a Feistel cipher, the text being encrypted is divided into two halves. The round function is applied to one half using a subkey and the output of F function is XORed with the other half. The two halves are then swapped. Each round follows the same pattern except for the last round where there is no swap. A nice feature of a Feistel cipher is that encryption and decryption are structurally identical, through the subkeys used during encryption at each round are taken in reverse order during decryption.

*3.1*InitialPermutation (P-Function)

The first step in the algorithm is the initial permutation function, which is applied before the first and after the last round. It is required to provide the necessary diffusion and confusion to the input block, where it's key dependent permutation such that additive differences will be destroyed as the key change. This could provide a protection against linear and differential cryptanalysis. The previous algorithm used XOR as a reversible mixing function before the first and after the last round. While the improved algorithm more complicated reversible mixing function is used before and after the last round. This would further confuse the entry values into the Feistel network and ensure a complete avalanche effect after the first two rounds.

Improved algorithm applies key dependent initial permutation as shown in Fig. (4). *P*-function has 128-bit input A and 128-bit output D. It Adopts “byte transposition” and the 40-bit subkey $(KP_1 | KP_2)$ to control data rotations.

Let $KP_1 = (m_1, m_2, m_3, m_4)$, and $KP_2 = (n_1, n_2, n_3, n_4)$, where mj and n_j 5-bit subkey and not equal to zero,

*

j= 1,…, 4. The function $D = P(A, KP_1 | \mathrm{KP}_2)$ is defined by following:

- Right rotation: $b_j = a_j >>> m_j$, for j=1,…,4.
- Byte transposition: $c_{jl} = b_{lj}$, for j, l=1,...,4.
- Left rotation: $d_j = c_j <<< n_j$, for j=1,…,4.

Each input word a_j affects all output words. In, *P*-function, the permutations are key dependent so that it could avoid linking plaintexts to input to the first *F*-function and ciphertexts to input to the last *F*-function. The *P*-function generates a number of permutations by using rotation and transposition. Where the generated group by rotating operation is isomorphic to cyclic group of order n. The number of permutations, which are generated by transposition, is equal to n! [2].

3.2 Encryption

The Encryption algorithm of the improved blowfish can be described as shown bellow. It is a Feistel network consisting of 16-rounds as described in Fig. (3).

Encryption algorithm

Input: A = plaintext M. {128-bit data element}.
Step1: D= P (A, P [0]). {P-function to performed initial permutation}
Step2: Divide D into two 64-bits XL, XR
Step3: For i =1, …, 16 do
Z = F (XL_i)
XR_i = P[i] XOR Z XOR XR_i
Swap XL_i and XR_i
Step4: Swap L_i and R_i undo the last round
Step5: Recombined A=(XL| XR}.
Step6: D=P (A, P[18]).
Output: Ciphertext D.

3.3 Function F

In the improved algorithm XL_i is divided into four 16-bit quarters a, b, c, and d (see Fig. (5)). The previous Blowfish algorithm uses each quarter as index to the one of the S-boxes, so that it uses four S-boxes while in the proposal all quarters use one common S-box. And consequently the number of S-boxes is reduced from four to one.

The entries are overlapped in a single S-box: entry 0 would consist of byte 0 through 7, entry 1 through 1 to 8, etc. This simplification would reduce the memory requirements for the four S-boxes from 2097152-byte (2^{16}×8 ×4) to single S-box of 65543 bytes. Additional steps will be required to eliminate the symmetries when different bytes of the input are equal, or when the 64-bits input to function F is a bitwise permutation of another 64-bits input. A more complex combining function in the proposed system is used to eliminate the symmetries (which will be described later in the next section).

Function F algorithm

Input: XL {the length of XL is 64-bits}
Set x1=0, x2=0, x3=0, x4=0
Set n1=0, n2=0, n3=0, n4=0
Step1: Divide XL into four 16-bits: a, b, c and d
Step2: For i =1 up to 8 Do
x1=x1 OR (S[a+i])
ROTATE_RIGHT (x1 8)
x2=x2 OR (S[b+i])
ROTATE_RIGHT (x2, 8)

*

x3=x3 OR (S[c+i])
ROTATE_RIGHT (x3, 8)
x4=x4 OR (S[d+i])
ROTATE_RIGHT (x4, 8)
End

Step3: n1= x1 AND 0xF
n2= x2>>4 AND 0xF
n3= x3>>8 AND 0xF
n4= x4>>16 AND 0xF
Y1=G (x1, n1)
Y2=G (x2, n2)
Y3=G (x2, n3)
Y2=G (x4, n4)

Z=(((Y1 + Y2) MOD 2^{64}) ^ Y3) + Y4) MOD 2^{64}

Output: Z {the length of Z is)

3.4 G-Function

In G-function (see Fig. 6), it uses rotations to achieve data permutation and multiplication to achieve diffusion. The function has 64 bits input and 64-bits output and the function D=G (X, N) is defined by following:

1.Divide X into four 16-bits quadratic a_1, a_2, a_3, a_4.

2. $k_j = a_j$ or 0x01, for *j*=1,…,4 (Let k_j be odd).

3. $c_1 = (k_1 <<< \mathbf{N})$ * k_4

4. $c_2 = (k_2 >>> \mathbf{N})$ * k_3

5. $c_3 = k_2$ * ($k_3 <<< \mathbf{N}$)

6. $c_4 = k_1$ * ($k_4 >>> \mathbf{N}$)

7. Combine $c_1, c_2, c_3,$ and c_4 in D.

The number of rotation N is different for each output of S-box because in each time it depends on the 4-bits, which truncate from variable position in each output. Hence we realize that if there are two or more similar inputs to the S-box there are no same outputs.

3.4 Subkeys

The range of values, which a key will take, became large. Where a large key space is necessary to prevent exhaustive search for a key (Solving the problem of finding the correct value for a key by testing possible values until the correct one is found).

The proposed system still uses the same key generation process because it is designed to preserve the entire entropy of the key and to distribute that entropy uniformly throughout the subkey. It is also designed to distribute the set of allowed subkeys randomly throughout the domain of possible subkey [6].

The improved algorithm uses:

1. The P-array of 18 64-bits subkeys:
2. $P_1, P_2, \ldots, P_{18}$.
3. There is one 8-bit S-box with 65543 entries: S_1, S_2, …S_{65543}.

The P-array and S-boxes must be precomputed before any data encryption or decryption. The same procedure, which is used in the previous Blowfish algorithm ,is used in this work to generate these subkeys.

4. Security of Improved blowfish algorithm

The most important requirement is stated succinctly in the AES announcement [1]: '*The security provided by an algorithm is the most important factor in the evaluation.'*

The improved Blowfish algorithm increased the security of the original Blowfish algorithm by using block size of 128-bits and allows 144 key lengths. The time-consuming subkey-generation process adds considerable

*

complexity for a brute-force attack algorithm.

The improved algorithm increased the number of iterations which will be required to test a single key. The complexity of algorithm is increased also by using a combination of basic operations. Hence by using *G*-function we achieve diffusion and confusion to the outputs of S-box. Each output of word d_j is effected by two words, where j=1, …, 4, So that there is no outputs of *G*-function are the same. On the other hand, the improved Blowfish algorithm increased the security by using the *P*-function, we note that each input word a_j affects all output words and each output word is affected by all input words. In, *P*-function, the permutations are key dependent so that it could avoid linking plaintexts to input to the first *F*-function and ciphertexts to input to the last *F*-function. The *P*-function and *G*-function use the combinations of basic operations to achieve a large number of encryption functions, i.e. permutations of binary n-bit vectors, high structural complexity.

4.1 An Attacker of the Improved Blowfish Algorithm:

Differential cryptanalysts work against block cipher algorithms that use constant S-boxes. The attack is heavily dependent on the structure of S-boxes; it looks specifically at ciphertext pairs whose plaintext has particular differences. The improved algorithm is patient to this type of attack and this belongs to many reasons:

- This type of attack is largely theoretical. The enormous time and data requirements to mount a differential cryptanalytic attack put it almost beyond the reach of everyone.
- Key-dependent permutation function is used before and after the last round such that the input bits are exchanged under the control of subkeys, so that the additive difference will be destroyed, as the bits are exchanged, this could provide protection against linear and differential cryptanalys is. 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 with 128-bits block size is resistant to matching ciphertext attack. It is required to 2^{64} ciphertext.

4.2 Avalanche Effect

In this section a statistical test on the ciphertext that produced from encryption the plaintext " ***This is a simple example of the block cipher system using Blowfish algorithm*****"** uses a key ***"block cipher algorithm"***.

Horst Feistel referred to the avalanche effect as: "*a small change in the key gives rise to a large change in the ciphertext".* [9]. Table (1) shows the avalanche effect on the plaintext when only one bit is changed in the key by using a block size of 64 bits and executing Blowfish algorithm before improvement.

Table (2) shows the avalanche effect on the same plaintext when only one bit is

*

changed in the same key by using a block size of 64 bits and executing Blowfish algorithm after improvement.

Table (3) shows the avalanche effect on the same plaintext when only one bit is changed in the same key by using a block size of 128 bits and executing Blowfish algorithm before improvement.

Tables 1, 2 and 3 are show that the changing are 24 to 38 bits out of 64 bits, to 43 bits out of 64 bits and 66 to 74 bits out of 128 bits when performing the algorithm before improvement, After performing the improved Blowfish algorithm respectively which mean that 37.5% to 59.5% of each block of the ciphertext is changed. After performing the improved Blowfish algorithm, the changing are 25 which mean that 39% to 67.2% of each block of the ciphertext is changed. However the changing are of the block size which mean that 51.6% to 58% of each block of the ciphertext is changed.

4.4 Complementation property

A block cipher satisfies the complementation property if for all plaintext blocks P and all keys K, are given:

C=E (P, K)

Then

C′=E′(P′, K′)

Where

C′: complement of C

P′: complement of P

K′: complement of K

H Gusafrson *et al.* state in their paper [9], that *" if a block cipher satisfies the complementation property this reduces the key search to half the number of keys possible in a chosen plaintext attack*". To show the complementation property on the improved algorithm, the key '***block cipher algorithm'*** to encipher a given plaintext. Two blocks size of 64 bits and 128 bits are used to encipher the same plaintext using the same key. After that the encipher algorithm is applied to the complement plaintext using the complement of the key. The complement property is not satisfied for improved Blowfish algorithm with block size 64-bits and 128-bits.

4.3 Memory requirement

One of the aims of the improved algorithm is to reduce the memory requirement without compromising security. Hence, the Blowfish's S-boxes are reduced to one S-box and the number of byte of this S-box is also reduced. Table (4), illustrates the memory requirement for the S-box to the Blowfish algorithm before and after improvement. It is clear from this table that the improved

The design philosophy of proposed algorithm depends on:

- Mixing operation from different algebraic group: XOR, addition, and multiplication.
- Using the subkey to control data permutation and data substitution.
- Reducing memory requirement without compromising security.

The proposed algorithm has the following features:

1. It uses the same algorithm criteria for encryption and decryption with the same key schedules, and supports variable key-length from 8 bytes up to 144 bytes.
2. It is based on simple theory principles and simple arithmetic operations and easy to implement. So, it is completely

*

specified, easy to understand the operation steps of the algorithm.

3. 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.
4. Each function of the proposed algorithm is dependent on its key, this wills preve nt fixed output and increase the non-linearity of the algorithm.

lowfish algorithm with 64-bits requires very small memory compared with the Blowfish with 64-bits.

5 Time requirement

Speed was not the primary goal of the AES competition. In this section, a comparison between the time requirements to encrypt 1 M bits using Blowfish algorithm before and after improvement.

Table (5) shows that the improved Blowfish algorithm is not as fast as the previous Blowfish algorithm, this disadvantage largely disappears when it is considered the likely platforms and applications of the 21st century.

Conclusions

The proposed Blowfish algorithm is a block cipher that encrypts data in 16-byte blocks. Key expansion converts a variable length key of 144 bytes into several sub key arrays totaling 65687 bytes. The proposed algorithm has 16 rounds. Its implementation and operation is very

so this will preve nt fixed output and increase the non-linearity of the algorithm.

depends on:

- Mixing operation from different algebraic group: XOR, addition, and multiplication.
- Using the subkey to control data permutation and data substitution.
- Reducing memory requirement without compromising security.

The proposed algorithm has the following features:

1. It uses the same algorithm criteria for encryption and decryption with the same key schedules, and supports variable key-length from 8 bytes up to 144 bytes.
2. It is based on simple theory principles and simple arithmetic operations and easy to implement. So, it is completely specified, easy to understand the operation steps of the algorithm.
3. 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.
4. Each function of the proposed algorithm is dependent on its key, so this will preve nt fixed output and increase the non-linearity of the algorithm.
5. Because of the large key and input block size of the proposed algorithm, exhaustive key search and the matching ciphertext attack are infeasible.
6. The small-number of bits to large-number of bits (to and from S-box) will have weaknesses with respect to linear cryptanalysis, but these weaknesses are hidden both by performing the G-function to the four outputs of S-box,

*

combining and making them depending on the key.

7. The improved Blowfish algorithm is not as fast as the previous Blowfish algorithm; disadvantage largely disappears with development of technology.

References

[1] Bruce Shnier "Applied Cryptography Second Edition Protocols. Algorithms, and Source, and Source Code in C", *John Wiley and Sons, Inc.*, 1996.

[2] Thilo Zieschang, " Combinatorial Properties of Basic Encryption Operations", Advances in Cryptology Eurocrypt'97, *International Conference on the Theory And Application of Cryptographic Techniques Konstanz, Germany, May 11-15, 1997 Proceedings, Springer*, 1997.

[3] Dieter Schmidt "*ABC - A Block Cipher*", Wikipedia the free Encycleopedia, May 27, 2002.

[4] Bruce Schneier and John *Kelsey," Unbalanced Feistel Networks and Block CipherDesign", Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419, 2005,*

[5] A. Bogdanov1, L.R. Knudsen, G. Leander1, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe," *PRESENT: An Ultra-Lightweight Block Cipher",* 2007, www.ist-ubisecsens.org

[6] Dr. Dobb's Journal, *September* 1995.

[7] *Fast Software Encryption Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag*, pp. 191-204 1994.

[8] Fred halsall " Multimedia Communications Applications, Networks, Protocols, And Standards, *ADDISON-WESLEY*, 2001.

[9] Shakir M. "A new feedback symmetric block cipher method", *Ph. D, Thesis Univ. of tech, Baghdad*, 1997.

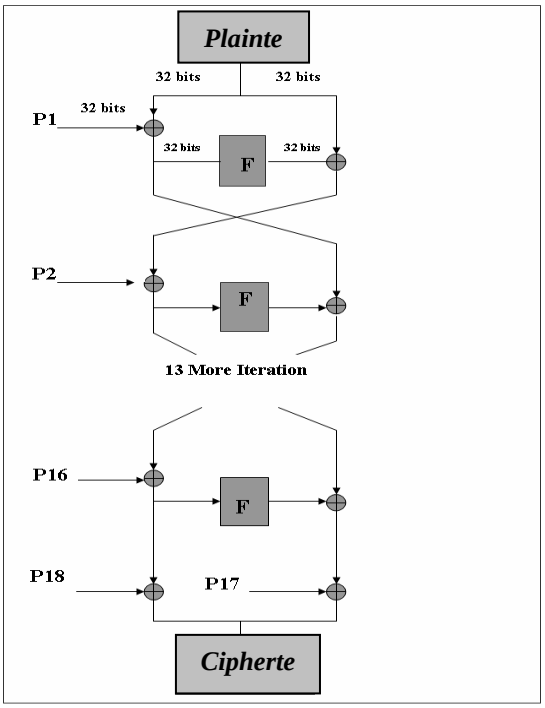

Fig (1) Blowfish Architecture

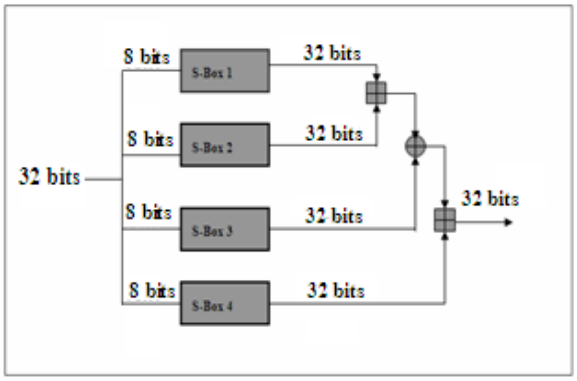

Fig. (2) Function F of the Blowfish algorithm

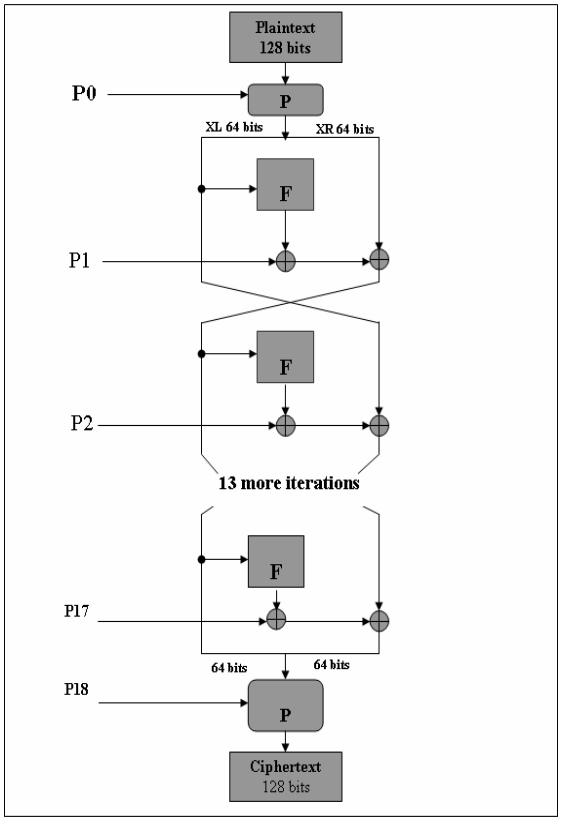

Fig. (3) Improved Blowfish Algorithm

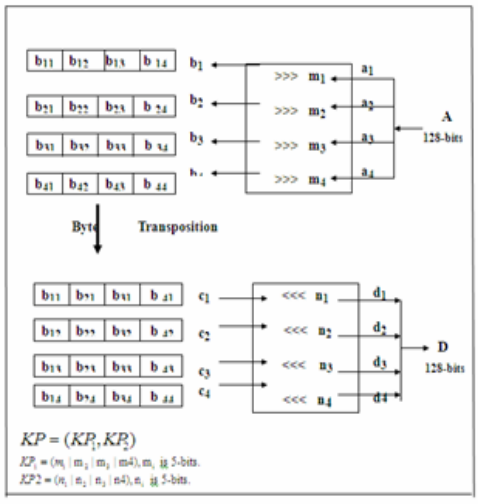

Fig. (4) P-function. $D = P(A, KP_1 | KP_2)$ for the Improved Algorithm

*

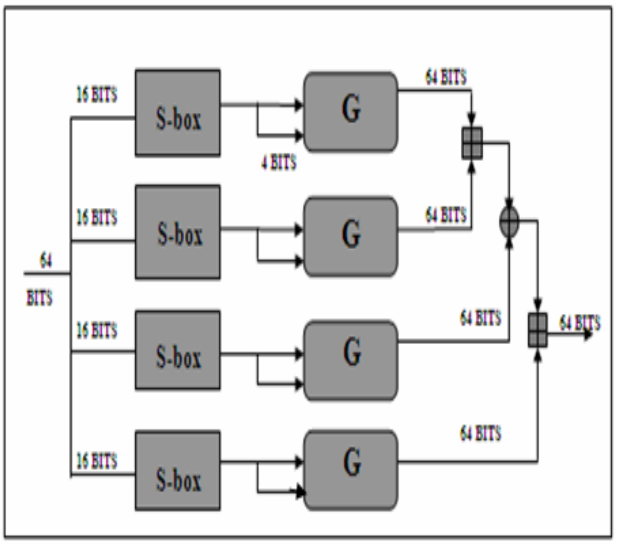

Fig.(5) Function F of the proposal Algorithm

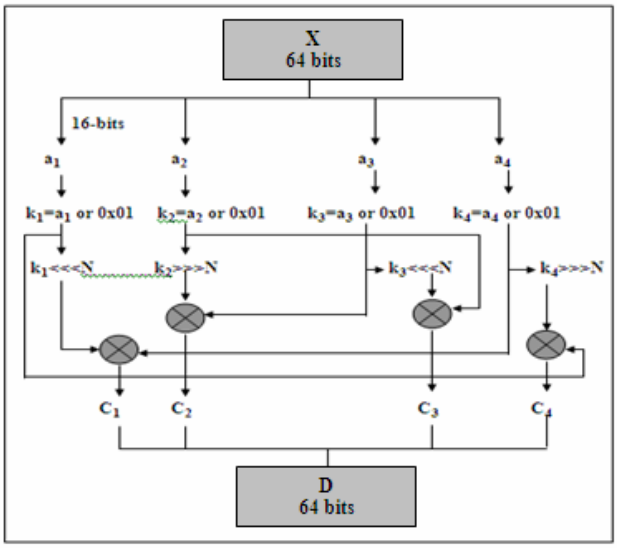

Fig. (6) Function-G of Proposal Algorithm

*

Table (1) Avalanche Effect of the original Blowfish with 64-bits block size: Change one bit in key

Block No.	Ciphertext 64-bits in Binary	Avalanche
1	0100100100001011110110101110100010010111100011010010110110001011 0001100010001011010110110111010010111101010000110110000110010010	24
2	1011101001011111110110110010100011101110001100110000100100111011 1111000001000110111011000100100010000110001101110111000111100000	27
3	1101110010100110010010010000100111010011110100001000001110110100 1111011000011101100001001000101011001001011010111100001001010100	31
4	1100000010100110010010010000100111010011110100001000001110110100 0111011000011111001100110100000110010111011011110100000100000100	32
5	0100001101100111000010011110110011001000001111001111010000010001 0010111001101010011111110011110110111111000110011000101001101001	36
6	0010001000010111000110011001101001101001010001000011010100001000 0011110100011011111001110111000011100110101110001110101110011000	38
7	0001110011100010010101101001001010011000011000000101100010011001 1 0111100010101100011000100010110111001110111111101110011000100001	33
8	1010110110110101011110001010010101100001011100101100010000101001 0101000001111110000000000110000011000101000010111110011010000010	35
9	0100101110001011010101010100110111000011011000001001110011011001 0001110110011111011010110110110111101010011101111000101010001010	26
10	1010110110111101011111100111101111110110100110010010101010000101 0 0100101110000010100101010000101000011010011010000101000001101011	37
Key1	block cipher algorithm	
Key2	alock cipher algorithm	

*

Table (2) Avalanche Effect of Improved Blowfish of 64-bits block size: Change one bit in key

Block No.	Ciphertext 64-bits in Binary	Avalanche
1	1100110010100100110011001010100101101111100001100100100010000001 1001001110110010101000001011100000010000110100100010110101010011	33
2	0100111110100011111000101011101110011010110101110101110100100010 0101000100101000001001011010010011111100110110001010110010100100	34
3	0010110100000100111000011110000011100011110001010010011110010000 0000100010111010111101011000000011110110110110011111111010011000	25
4	1000001001111000010111110011111000000111000010011111010111011100 1000101111000101100101001101011101110100110001111011010100111100	32
5	0001100010110110111000110111010011011111110000101010010011100100 0010101110000000000000000011010101010110100000101100110000100101	35
6	0111000011010100011110110100111010010101001100011010110101001100 1010110100000101100110000010010111010101110000111000010101100111	32
7	0000100101110010010000111000111100000011111001011000110100101001 0001001110111111110110010001111001101111000001111011100101000110	43
8	0101011001011111001111011111111110100011111000101001001111011001111 0110111010100011110110111101100100101111011111101011011010001000	31
9	1011011101110111011100100001000001010011001110101000111001100100 1110010000011101100000111010011000101010101010000011000110010110	38
10	0001101010100010111000000001111011001111000111111010111101010110 0101011011000010011011110101011110001101111101110101101010111011	32
Key1	block cipher algorithm	
Key2	alock cipher algorithm	

*

Table (3) Avalanche Effect of Improved Blowfish of 128-bitsblock size: Change one bit in key

Block No.	Ciphertext 128-bits in Hexadecimal	Avalanche
1	0xba021566600b2efaa057aaf6a1411123 0xcb52ca63c5f6c4c8dd059d35f33e1225	66
2	0xf8d2cad810becf636dc3cbbf079c1d3a 0x5e3559c5c89e79d4fe66b857bebce685	74
3	0x39c66217cfa6ba516a2288cff8a093c7 0x5e796c88dca76e18875827420b625440	71
4	0x587a661c67d3903fee95536ba07825d1 0xca1ba7b4fb66d4609422250bdb3c8343	68
5	0xcb52ca63c5f6c4c8dd059d35f33e1225 0x8980b69a9203489cc7e022168ca4f047	68
Key1	block cipher algorithm	
Key2	alock cipher algorithm	

Table (4) Memory requirement comparisons of S-box

Algorithm	Block size (bits)	Memory requirement (byte)
Blowfish	64	4096
Blowfish	128	2,097,152
Improved Blowfish	64	259
Improved Blowfish	128	65,543

Algorithm	Block size (bits)	Speed M bit/sec
Blowfish	64	0.3
Improved Blowfish	64	4.6
Improved Blowfish	128	3.8

Table (5) Speed Comparisons of Blowfish Algorithm on a Pentium II

*