Type-3 Feistel Network of The 128-bits Block Size Improved Blowfish Cryptographic Encryption

Ashwaq T. Hashim*

Received on: 28/5/2008
Accepted on: 6/11/2008

Abstract

In this paper, a new secret-key block cipher called 128-bits Blowfish is proposed which is an evolutionary improvement of 64-bits Blowfish designed to meet the requirements of the Advanced Encryption Standard (AES) to increase security and to improve performance. The proposed algorithm will be used a variable key size up to 192 bytes. It is a Type-3 Feistel network iterated simple function 16 times. Specifically, in this proposed algorithm, a combination of four S-boxes lookups, multiplications as well as fixed and data-dependent rotations will be used. Mixing round provide different levels of security, efficiency, flexibility and good avalanche effect. The proposal is word-oriented, in that all the internal operations are performed on 32-bit words. So it is possible to implement the algorithm on smart cards efficiently.

Keywords: Blowfish, cryptography, Feistel Network, block cipher, AES.

شبكة فيستيل نوع 3 وحجم كتلة bits-128 لخوارزمية التشفير المطورة Blowfish تشفير

الخلاصة

في هذا البحث ,اقترحت خوارزمية تشفير كتلية ذات المفتاح السري مسماة 128-bits Blowfish وهو تحسين تطويري الى 64-bits Blowfish صممت لتحقيق متطلبات معيار التشفير المتقدم(AES) لزيادة الامنية وتحسين الاداء. الخوارزمية المقترحة سوف تستخدم مفتاح متغيّر يقدّر حجمه إلى حد 192 بايت. هو شبكة فيستيل نوع 3 يكرر وظيفة بسيطة 16 مرة . وخاصة فأن الخوارزمية المقترحة تستخدم مجموعة من اربعة صناديق , ضرب ودورات ثابتة ومعتمدة على بيانات. . توفر دورات الخلط مستويات مختلفة من الامنية ,الكفاءة , المرونة وavalanche جيد . ان الخوارزمية المقترحة هي word-oriented حيث كل العمليات الداخلية تنفذ على 32-bit words لذلك من الممكن تطبيق الخوارزمية على البطاقات الذكية (smartcard) بكفاءة.

***Control and Systems Department, University of Technology / Baghdad**

https://doi.org/10.30684/etj.27.2.2
2412-0758/University of Technology-Iraq, Baghdad, Iraq

1. Introduction:

Symmetric-key block ciphers have long been used as a fundamental cryptographic element for providing information security. Although they are primarily designed for providing data confidentiality, their versatility allows them to serve as a main component in the construction of many cryptographic systems such as pseudorandom number generators, message authentication protocols, stream ciphers, and hash functions. There are many symmetric-key block ciphers which offer different levels of security, flexibility, and efficiency. Among the many symmetric-key block ciphers currently available, some (such as DES, RC5, CAST, Blowfish, FEAL, SAFER, and IDEA) have received the greatest practical interest [1].

Most symmetric-key block ciphers (such as DES, RC5, CAST, and Blowfish) are based on a "Feistel" network construct and a "round function". A Feistel cipher involves dividing the plaintext into two halves (the length of them depend on the algorithm which is used) and repeatedly applying a round function to the data for some number of rounds, where in each round using the round function and a key (K1, K2,….Kn where n represent number of round), the left half is transformed based on the right half and then the right half is transformed based on the modified left half as shown in figure (1). The round function provides a basic encryption mechanism by composing several simple linear and nonlinear operations such as Exclusive-OR, substitution, permutation, and modular arithmetic [2].

Different round functions provide different levels of security, efficiency, and flexibility. The strength of a Feistel cipher depends heavily on the degree of diffusion and non-linearity properties provided by the round function. Many ciphers (such as DES and CAST) base their round functions on a construct called a "substitution box" (s-box) as a source of diffusion and non-linearity. Some ciphers (such as RC5) use bit-wise data-dependent rotations and a few other ciphers (such as IDEA) use multiplication in their round functions for diffusion [1].

This paper present some existing modern cipher such as ABC is 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].

In this paper, an improvement of Blowfish algorithm which is symmetric-key block cipher is presented with a block size of 128 bits and a variable key size, up to 192 bytes. The philosophy of proposal algorithm is to use the full menu of "strong operations" supported in modern computers to achieve better security properties and provide high speed. The main aim behind the design of this proposal is to get the best security/performance tradeoff over existing ciphers.

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,……., P_{18}.

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

$S_{1,0}$, $S_{1,1}$,…., $S_{1,255}$;

$S_{2,0}$, $S_{2,1}$,…., $S_{2,255}$;

$S_{3,0}$, $S_{3,1}$,…., $S_{3,255}$;

$S_{4,0}$, $S_{4,1}$,…., $S_{4,255}$;

2.2 Encryption and Decryption:

The underlying philosophy behind Blowfish is that simplicity of design yields algorithm that is both easier to implement and through the use of a streamlined Feistel network, 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 (2) 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, 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.

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 permit key lengths up to 512 bits.

2.3 Function F:

The function F, shown in Fig.3 can be described as follows [6]. Divide XL into four eight-bit quarters: a, b, c, and d. Then,

$$F(XL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}. \quad \ldots\ldots(1)$$

The non-reversible function is designed for strength, speed, and simplicity. The function that combines the four S-box outputs must be fasted 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 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 Proposed Algorithm:

The proposed algorithm works with four 32-bit registers A;B;C;D which contain the initial input plaintext as well as the output ciphertext at the end of encryption. The cipher is working with 32 bit words in that all the operations in proposal algorithm are applied to 32-bit word. This algorithm is a type-3 Feistel network iterated simple function 16 times (Fig. 4). Mixing round provide different levels of security, efficiency, flexibility and good avalanche effect. (See section 4).

The Proposed algorithm is designed to use a full menu of "strong operations" supported in modern computers to achieve better security properties, high speed, and implementation flexibility. Proposal algorithm will be used primitive operations (add, subtract, multiply, exclusive-or, and data-dependent rotate). The proposal algorithm is resistance against linear and differential attacks by using four key dependent (S-box) tables of 255 32-bit which is also good avalanche of data and key bits.

The Algorithm, shown in Fig. 5, shows the encryption operation of proposed algorithm in details. The cipher is working with 32 bit words in that all the operations, which are viewed as unsigned integers. In this algorithm the following notations are used. The denoted by c^d a bitwise Exclusive-OR of the two words c and d and the denoted by c+d addition modulo 2^{32}, by c-d subtraction modulo 2^{32}, and by c×d multiplication modulo 2^{32}. Also, c<<<d and c>>>d, denote cyclic rotations of the 32-bit word c by d positions to the left and right, respectively.

In each round the output of F-function is the input to E-function (which is derived from MARS algorithm) then, one data word will be used as the input to the E-function and the three output words from the E-function are added or XORed to the other three data words. In addition, the source word is rotated by 13 positions to the left [1]. The proposal uses the same structure of F-function of previous Blowfish algorithm.

The decryption operation of proposed algorithm is the inverse of the encryption operation and the code for decryption is similar.

3.1 The E-function:

As mentioned above the E-function is derived from Mars Encryption algorithm [1] with some

modifications in proposal algorithm. The E-function takes as input one data word and uses two more key words to produce three output words. In this function three temporary variables will be used, denoted below by L, M and R (for left, middle and right). Below it is also refer to these variables as the three "lines" in the function. Initially, R will be set to hold the value of the source word rotated by 13 positions to the left, and M will be set to hold the sum of the source word and the first key word. Then the lowest nine bits of M will be viewed as an index to S-boxes (the S-box is chosen by using this function (*number of round mod 4*), and set L to hold the value of the corresponding S-box entry.

The second key word is multiplied (constrained to contain an odd integer) into R and then rotate R by 5 positions to the left (so the 5 highest bits of the product becomes the 5 lowest bits of R after the rotation). Then R XORed into L, and also view the five lowest bits of R as a rotation amount between 0 and 31, and rotate M to the left by this amount. Next, R will be rotated R by 5 more positions to the left and XOR it into L. Finally, again the five lowest bits of R will be viewed as a rotation amount and rotate L to the left by this amount. The first output word of the E-function is L, the second is M and the third is R (see figure 6).

The Algorithm in Figure 7 shows the E-function operation of proposal algorithm in detail**.**

3.1.1 Design rationale

In the design of the E-function a combination of different operations will be used in a way that would maximize the advantages from each. Some properties of this function which are the following:

a) Recall that when two words are multiplied, the lower bits of the input word have larger effect on the product than the higher bits. Thus, it is arranged so that bits which are *not fed as input to the S-box* will be the lowest bits in the data word which is being multiplied. The amount of rotation (13 bits) was set to maximize the resistance of the E-function to differential attacks. Also, since the internal structure of the E-function is very sensitive to the location of the input bits, it makes sense to apply a constant rotation to the data lines, so as to make it hard for an attacker to maintain a consistent behavior across rounds [1].

b) Recall also that when two words are multiplied, the most significant bits in the product are the "stronger bits" since they are affected by almost all the input bits. In the combination of the multiplication and the data-dependent rotation, therefore it is arranged so that these "strong bits" are used to determine the amount of the data-dependent rotation [1].

c) Since the E-function is supposed to approximate a pseudo-random function, we would like to make the three lines of the function as "independent of each other" as possible. Thus very little interaction between the data in the three lines will be used. This also helps to avoid unwanted cancellations and makes it harder to obtain a linear approximation of one line in terms of another.

d) Still trying to guarantee some measure of "independence" between the data lines, the value of one line will make sure that never completely determines the value of another line. Indeed, the relative entropy of any two lines is at least 9 bits (of lines L; R), and gets as high as 32 bits (of lines R; M).

e) Since Line M will be viewed as the weakest output of the E-function (as it only carries the sum of the input and a key word, rotated by some amount), it is putted as the middle output line. In

this way, it never affects the next data line which is used as a source, but rather a data line which is used further down in the encryption process.

3.2 Subkey of proposed algorithm

Two basic ways ensure that the key is long enough to ensure a particular security level. One is to carefully design the algorithm so that the entire entropy of the key is preserved, so there is no better way to crypanalyze the algorithm other than brute force. The other is to design the algorithm with so many keys so that attacks that reduce the effective key length by several bits are irrelevant. The range of values, which a key will take, became large. 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) [8]. The proposed will increase the maximum key length from 64 bytes, in previous Blowfish algorithm, to 192 bytes.

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 [7]. So P-array of 48 32-bits subkeys: P_1, P_2, ……..…, P_{48} are used·

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

4. Security of Proposed Blowfish algorithm

The most important requirement is stated succinctly in the AES announcement [9]: '*The security provided by an algorithm is the most important factor in the evaluation.'*

The security obtained by using the proposed algorithm is increased, compared the original Blowfish algorithm, by using block size of 128-bits and allows 192bytes key length. The time-consuming subkey-generation process adds considerable complexity for a brute-force attack.

The complexity of proposal algorithm will be increased by using combinations of basic operations. Hence by using *E*-function a diffusion and confusion to the outputs of S-box will be achieved. The permutations in proposed algorithm 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. An *E*-function will be used a 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 proposal 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. It analyzes the evolution of these differences as the plaintext propagates through the rounds of algorithms when they are encrypted with the same key. Certain differences in plaintext pairs have a high probability of causing certain differences in the resulting ciphertext pairs. The proposal 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 in each 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 exchange, this could provide protection against linear and differential cryptanalysis. Linear cryptanalysis is work with large S-box (which is key dependent) and mapping is small number of bits to large number of bits. Using *E*-function to the outputs of the S-box and making this function dependent on subkey hides these weaknesses.

Data dependent rotations can be performed quickly in software and hardware. Combined with arithmetic operations (such as addition), this operation is very effective against linear cryptanalysis. One problem with data-dependent rotations is that specifying a rotation amount for a w-bit word only takes log w bits. Hence, while the result of this operation depends on all the bits in one operand, it only depends on very few bits in the other. This may lead to differential weaknesses, as was recently demonstrated by Biryukov and Kushilevitz [10].

4.2Avalanche Effect

Horst Feistel referred to the avalanche effect as: "*a small change in the key gives rise to a large change in the ciphertext"*. [11]. Table 1 shows the avalanche effect on the plaintext when only one bit is changed in the key by using Blowfish, RC6, Serpent and proposal algorithm. For example if the plaintext **X=AAAAAAAAAAAAAAAA** (16 characters (128 bits)) will be encrypted by proposal algorithm using two key which are changed in one bit resulting two ciphertext represented in hexadecimal:

Y1=7241A8D34A145F36135109ED A679B50C

Y2=E4B58EF0C581E19E68C0FEB 437A797C5

The avalanche effect computed by finding the number of changed in bits between these two ciphertext in above case it is equal to 68.

This section is prepared for making statistical test on the ciphertext that produced from encryption the plaintext:

1. **AAAAAAAAAAAAAAAA**
2. **BBBBBBBBBBBBBBBB**
3. **DDDDDDDDDDDDDDDD**
4. **EEEEEEEEEEEEEEEE**
5. **FFFFFFFFFFFFFFFF**
6. **HHHHHHHHHHHHHHHH**
7. **IIIIIIIIIIIIIII**
8. **NNNNNNNNNNNNNNN**
9. **MMMMMMMMMMMMMMM**
10. **PPPPPPPPPPPPPPP**

Table (1) shows that the changing are 24 to 35 bits out of 64 bits of the block size in Blowfish algorithm, while the changing are 60 to 72 bits, 57 to 71 bits and 59 to 73 bits out of 128 bits block size in RC6, Serpent and the proposed algorithm respectively. Moreover the average of avalanche effect of Blowfish algorithm is 30.9%. The averages of the avalanche effect are 65.5, 64.8 and 67 of RC6, Serpent and Proposal algorithm respectively. Where the number of block that has weights greater than half of the block length are 6, 6, 6, and 9 in Blowfish, RC6, Serpent and the proposed algorithms respectively.

5. Time comparison

In this section the time requirements are computed for Blowfish, RC6, Serpent and Proposed Algorithm. RC6 and Serpent are two algorithms of final list candidates. RC6 is the fastest one but the Serpent is lowest among them. So that these algorithm are chose to compare with proposal algorithm and with the original Blowfish algorithm. Table (2) shows these comparisons:

6. Conclusions

A secure, compact and simple block cipher algorithm is proposed. It offers good performance

a considerable exibility. Furthermore, its simplicity will allow analysts to quickly refine and improve our estimates of its security. It offers much improved security/performance over previous Blowfish algorithm by taking advantage of the powerful operations supported in today's computers.

During the design process, several things can be concluded about cipher design:

1. The Proposed algorithm is designed to be used in upgraded computer environments. It uses the full menu of "strong operations" supported in modern computers to achieve better security properties. This approach enables us to get better security per-instruction ratio for our software implementation of this proposal than is possible for existing ciphers. The design takes full advantage of the ability of today's computers to perform fast multiplications and data-dependent rotations.
2. Working with 32 bit words. Since most computers today (and in the near future) use word-size of 32 bits, all the operations in proposal algorithm are applied to 32-bit words. At the current state of the technology, this choice provides a good tradeoff between the ability to run the algorithm on computers which are available today (as well as on legacy systems and even 8-bit processors), and the ability to take advantage of larger word-size in future architectures.
3. Type-3 Feistel network. Since proposed algorithm has a block length of 128 bits and word-size of 32 bits, it follows that each block consists of four words. Among the various network-structures which are capable of handling four words in a block, it seems that a type-3 Feistel network provides the best tradeoff between speed, strength and suitability for analysis. A type-3 Feistel network consists of many rounds; where in each round one data word (and a few key words) is used to modify all the other data words. Compared with a type-1 Feistel network (where in each round one data word is used to modify one other data word), this construct provides much better diffusion properties with only a slightly added cost. Hence, fewer rounds can be used to achieve the same strength. Additionally, a type-3 Feistel network has advantages over structures in which several data words are used "at once" to modify other data words, in that these structures are typically much harder to analyze (and hence, much more prone to design errors). The reason is that in such structures the analysis must take into account all the possible combinations of values for the input data words, which quickly leads to unmanageable complexity.
4. Symmetry of encryption and decryption. We designed MARS to be as secure against chosen ciphertext attacks as against chosen plaintext attacks. This dictates making the cipher very symmetric, so the last half of the rounds are almost a "mirror image" of the first half.
5. From the results that were obtained in section 4 and after measuring the strength of the proposed algorithm it can conclude that the proposal algorithm increases the security and complexity compared with RC6, Blowfish and Serpent algorithms.

7.References

[1] Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O'Connor, M. Peyravian, D. Safford and N. Zunic, "*Mars a candidate cipher for AES*", *First Advanced Encryption Standard (AES) Conference*, Ventura, CA, 1998.

[2] Feistel H., "*Cryptography and Computer Privacy*," Scientific American, Vol. 228, no. 5, May 73, pp. 15-23.

[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 Cipher Design", Counterpane* Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419, 2005,*http://fschneier,kelseyg@counterpane.com*

[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] Bruce Schneier**,"** *The Blowfish Encryption Algorithm -- One Year Later*", Dr. Dobb's Journal, *September*

[7] Bruce Schneier " *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)* ", Fast Software Encryption Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, pp. 191-204 1994.

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

[9] Ross Anderson, Eli Biham, and Lars Knudsen, "Serpent*: A Proposal for the Advanced Encryption Standard*", *An Internet Survey*, 2000.

[10] A. Biryukov and E. Kushilevitz, "Improved cryptanalysis of RC5", *Advances in Cryptology, EUROCRYPT 98, Lecture Notes in Computer Science, vol. 1403*, K. Nyberg ed., Springer- Verlag, pages 85–99,2000. *http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Serpent/Serpent.pdf.*

[11] Shakir M. "A *new feedback symmetric block cipher method*", Ph.D. Thesis University of Technology, Baghdad, 199

Table (1) Avalanche Effect of Blowfish, RC6, and Serpent and the Proposed Algorithm: Change one bit in key

Block No.	Blowfish	RC6	Serpent	Proposal Algorithm
1	32	62	68	68
2	32	63	62	64
3	29	71	61	66
4	35	61	70	70
5	33	72	71	59
6	33	64	57	71
7	24	68	65	66
8	31	66	65	67
9	32	60	67	73
10	28	69	62	66
Avg.	**30.9**	**65.6**	**64.8**	**67**
Key 1	**00000000000000000000000000000000**			
Key 2	**00000000000000000000000000000001**			

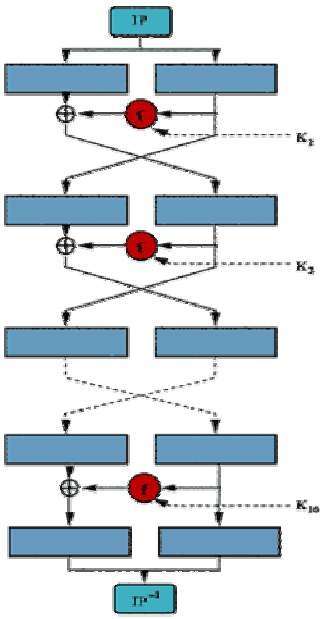

Figure (1) General design of a Feistel cipher

Table (2) Time Comparisons of Blowfish, RC6, Serpent and Proposal on a Pentium I4

Algorithm	Block size In bits	Number of Round	No. of Bytes	Time in Second
Blowfish	64	16	1000	0.01
			10000	0.1
			100000	0.35
RC6	128	20	1000	0.04
			10000	0.23
			100000	1.37
Serpent The	128	32	1000	1.57
			10000	14.56
			100000	150.01
Proposed Algorithm	128	16	1000	0.06
			10000	0.2
			100000	1.98

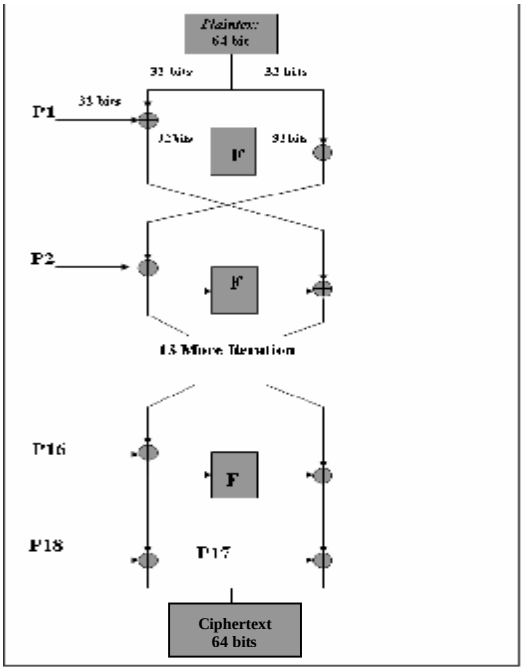

Figure (2) Blowfish Architecture

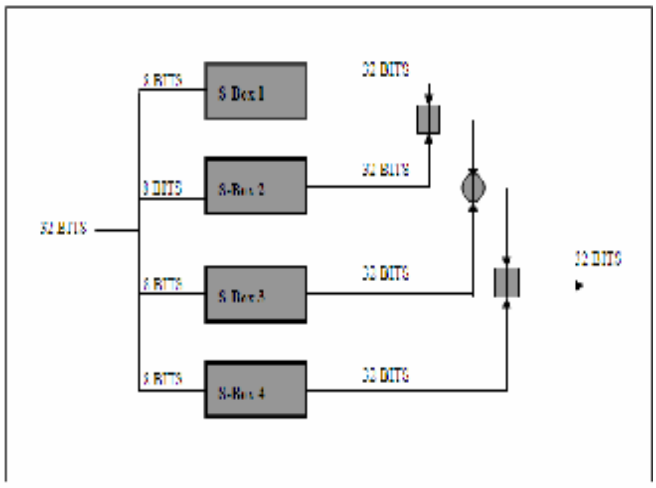

Figure (3) Function F of the Blowfish algorithm

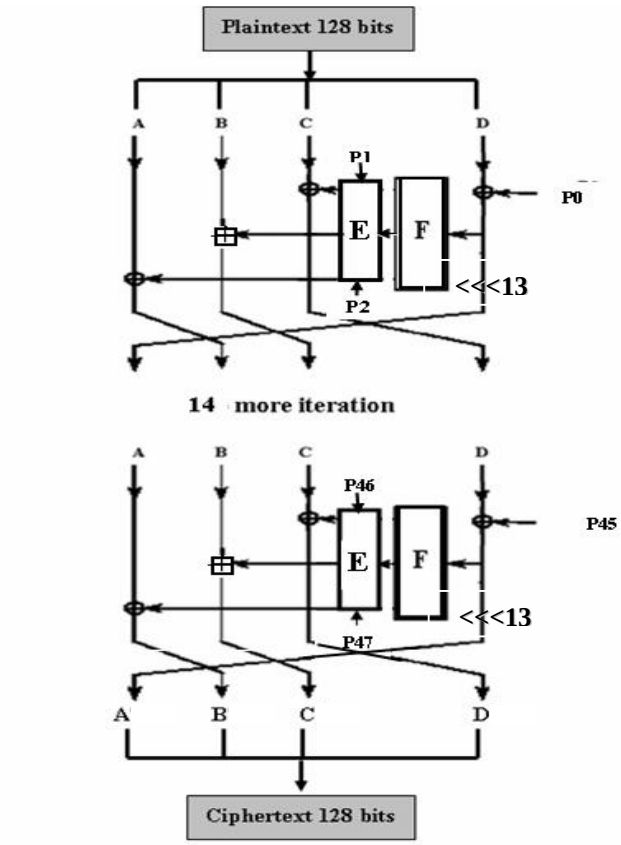

Figure (4) The Proposal Algorithm

Encryption:
Input: Plaintext = (A, B, C, D) // *each of which is 32 bits*
Output: Ciphertext= (A, B, C, D) // *each of which is 32 bits*
Procedure:
For i = 0 to 15 do
{
D=D^ P [i*3]
in=F (D)
D=D<<<13
E (in, P [i*3+1], P [i*3+2], L, M, R); // *output (L; M; R)*
C=C^L;
B=B+M
A=A^R
(A, B, C, D)= (B, C, D, A)
}

Figure (5) The Pseudo-code of Encryption for Proposed Algorithm

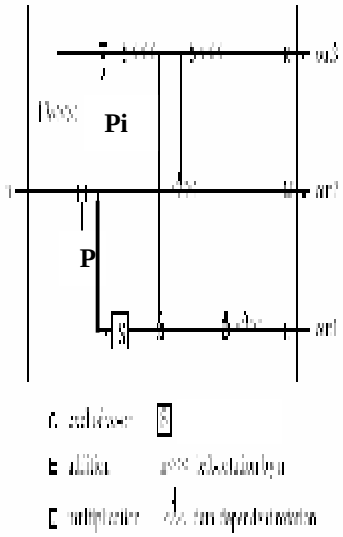

Figure (6) The E-function

E-function
Input: in; key1; key2
Output: L; M; R
Producer:
M = in + key1 //add first key word
R = (in<<< 13)×key2 // multiply by 2nd key word, *which must be odd*
i = lowest 9 bits of M
L = S[i mod 4] // S-box lookup
R = R <<< 5
r = lowest 5 bits of R // these bits specify rotation amount
M = M <<< r // 1st data-dependent rotation
L = L ^R
R = R<<< 5
L = L^ R
r = lowest 5 bits of R // these bits specify rotation amount
L = L <<< r // 2nd data-dependent rotation

Figure (7) The Pseudo-code of E-function