Src: Hybrid Block Cipher Method

Ashwaq T. Hashim

Received on: 5/9/2005
Accepted on: 26/1/2006

Abstract

The principal goal guiding the design of any encryption algorithm must be the security. In the real world, however, performance and implementation cost are always of concern. Making the assumption that the major AES candidates are secure the most important properties the algorithms will be judged on will be the performance and cost of implementation.

In this paper, a new secret-key block cipher called SRC, is proposed. The block size is 128 bits, and the key can be any length up to 256 bytes. The proposed algorithm is designed to take advantage of the powerful, which is supported by RC6 and Serpent algorithms with overcoming their weaknesses, resulting in a much improved security/performance tradeoff over existing ciphers. As a result, this proposed algorithm offers better security than RC6 and Serpent algorithms. And it is compact, speed, simplicity and easy to understand. It adapts key dependent permutation and combines different algebraic group which is adapted from RC6 algorithm. So that the SRC algorithm will be combined the Serpent and RC6 algorithms to obtain security/performance tradeoff and fastest algorithm.

الخلاصة

الهدف الرئيسي الذي يقود الى تصميم اية خوارزمية تشفير يجب ان يكون الامنية في العالم الحقيقي ، على اية حال فأن تكاليف الاداء والتطبيق هي التي تهم دائما " لنفترض بأن الانواع الرئيسية ل (AES) تمتلك الامنية ، فأن اهم الخصائص التي سوف تتحكم في الخوارزميات سوف تكون الاداء وتكاليف التطبيق .
في هذا البحث فأن هناك اقتراح باستعمال طريقة جديدة للتشفير الكتلي ذات المفتاح السري تدعى (SRC) . ان حجم الكتلة هو ١٢٨ bits وان طول المفتاح غير محدد وقد يصل الى ٢٥٦ bits كحد اعلى . الخوارزمية المقترحة مصممة بحيث تأخذ مزايا القوة من خوارزميات التشفير الكتلي RC6و Serpent وهي كذلك تتصف بكونها مدمجة ، سريعة ، بسيطة وسهلة الاستعمال . فانها تتبنى التبديلات المعتمدة على المفتاح وتدمج مجموعات جبرية مختلفة والتي تبنتها من خوارزمية RC6 . ان خوارزمية Serpent تمتلك امنية عالية ولكن تطبيق احسن برمجيات ابطأ ٣-٤ مرات من الاخريات وبضمنهم (Rijndeal ، Serpent ، RC6 ،Mars ، and Twofish) . من الناحية الاخرى فأن خوارزمية RC6 هي اسرع خوارزمية مقارنة بالاخريات وبهذا فأن خوارزمية SRC المحسنة سوف تجمع الخوارزميات RC6 و Serpent للحصول على خوارزمية ذات أمنية قوية وسرعة عالية .

Introduction

The SRC cipher uses a variety of operations to provide a combination of high security, high speed, and implementation flexibility. The main theme behind the design of SRC is to get the best security/performance tradeoff by utilizing the strongest techniques available today for designing block ciphers.

University of Technology
Control and System Eng. Dept.

In this paper, a symmetric-key block cipher, called SRC will be presented, with a block size of 128 bits and a variable key size, 256 bytes. The SRC algorithm uses an integer multiplication, rotation and shift as additional primitive operations which are greatly increases the diffusion achieved per round, allowing for greater security, fewer rounds, and increased throughput as a result the new SRC has much faster diffusion than Serpent.

The design of the SRC began with a consideration of RC6 and Serpent as a potential candidate for an AES $\mathbf{K_0}$, ..., $\mathbf{K_r}$. The user key can be any length between 64 and 256 bits. Short keys with less than 256 bits are mapped to full-length keys of 256 bits by appending one "1" bit and as many "0" bits are required to make up 256 bits [2].

- The cipher itself as an SP (Substitution -Permutation) - network and consists of:
- An initial permutation IP.
- 32 rounds, each consisting of a key mixing operation, a pass S-boxes, and (in all but the last round) a linear transformation. In the last round, this linear transformation is replaced by additional key mixing operation.
- A final permutation FP.

Each round function $\mathbf{R_i}$ {i=0,..., 31} uses only a single replicated S-box. For example, $\mathbf{R_0}$ uses $\mathbf{S_0}$, 32 copies of which are applied in parallel. Thus the first copy of S_0 takes bits 0, 1, 2 and 3 of $B_0 \oplus \mathbf{K_0}$ as its input and returns as output the first four bits of an intermediate vector; the next copy of S_0 input bits 4-7 of $\mathbf{B_0} \oplus \mathbf{K_0}$ and returns the next four bits of the intermediate vector, and so on. The intermediate vector is then transformed using the linear submission. Modifications were then made to meet the AES requirements. The outer loop, however, is based around the same found in Serpent. SRC will be intentionally designed to be extremely simple, to invite analysis shedding light on the security provided by extensive use of key dependent permutation like RC6.

2. Serpent Algorithm

2.1 Encryption and Decryption

Serpent encrypts a 128-bit plaintext P to a 128-bit ciphertext C in $\mathbf{r}$ rounds under the control of $\mathbf{r+1}$ subkey transformation, giving $\mathbf{B_1}$. Similarly, $\mathbf{R_1}$ uses 32 copes of $\mathbf{S_1}$ in parallel on $\mathbf{B_1} \oplus \mathbf{K_1}$ and transforms their output using the linear transformation, giving $\mathbf{B_2}$.

In the last round $\mathbf{R_{31}}$, $\mathbf{S_{31}}$ is applied on $\mathbf{B_{31}} \oplus \mathbf{K_{31}}$, and XOR the result with $\mathbf{K_{32}}$ rather than applying the linear transformation. The result $\mathbf{B_{32}}$ is then permuted by FP, giving the ciphertext.

As with DES, the final permutation is the inverse of the initial permutation. Thus the cipher may be formally described by the following equation:

$$B_0 = IP(P)$$

$$B_{i+1} = R_i(B_i)$$

$$C = FP(B_r)$$

Where

$$R_i(X) = L(S_i(X_i \oplus K_i)) \quad i = 0, \ldots, r\text{-}2$$

$$R_i(X) = S_i(X \oplus K_i) \oplus K_i \quad i = r\text{-}1$$

Where $\mathbf{S_i}$ is the application of the S-box $\mathbf{S_i}$ 32 times in parallel, L is the linear transformation (shows later).

Decryption is different from encryption in that the inverse of the S-boxes must be used, as well as the

inverse linear transformation and reverse order of the subkeys.

2.2 The Key Schedule

The serpent encryption requires 132 32-bits of key material. First the user supplied 256-bit key **K** is expanded to 33 128-bit subkeys $K_0,\ldots, K_{32}$, the key **K** will be written as eight 32-bit $w_{i-8}, \ldots w_{i-1}$ and expand these into an intermediate key $w_0, \ldots, w_{131}$ by the following affine recurrence[3]:

$$w_i=(w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \emptyset \oplus i) <<< 11$$

Where $\emptyset$ is the fractional part of the golden ratio 0x9e3779b9 in hexadecimal. The underlying polynomial $X^9+x^7+x^5+x^3+1$ is primitive, which together with the addition of the round index is chosen to ensure an even distribution of key bits throughout the rounds, and to eliminate weak keys and related keys.

The round keys are now calculated from the prekeys using the S-boxes. The S-boxes is used to transform the prekeys w_i into words k_i of round key by dividing the vector of prekeys into four sections and transformation the i^{th} words of each of the four sections using $S_i(r+3-i) \bmod r$. This can be seen simply for the default case **r=32** as follows:

$$\{k_0, k_{33}, k_{66}, k_{99}\} = S_3\{w_0, w_{33}, w_{66}, w_{99}\}$$

$$\{k_1, k_{34}, k_{67}, k_{100}\} = S_2\{w_1, w_{34}, w_{67}, w_{100}\}$$

...

$$\{k_{31}, k_{64}, k_{97}, k_{130}\} = S_4\{w_{31}, w_{64}, w_{97}, w_{130}\}$$

$$\{k_{32}, k_{65}, k_{98}, k_{131}\} = S_3\{w_{32}, w_{65}, w_{98}, w_{131}\}$$

After that renumber the 32-bit values k_j as 128-bit subkeys K_i *(for* $i \in \{0...r\}$*)* as follows: $K_i=\{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\}$

The ***IP*** is applied to the round key in order to place the key bits in the correct column, i.e., $\mathbf{K_i} = IP(K_i)$. Figure 3 illustrates the key generation algorithm [2].

3 The RC6 Algorithm

RC6 is a fully parameterized family of encryption algorithms [3]. A version of RC6 is more accurately specified as RC6-w/r/b where the word size is w bits, encryption consists of a nonnegative number of rounds r, and b denotes the length of the encryption key in bytes. Since the AES submission is targeted at w = 32 and r = 20.When any other value of w or r is intended in the text, the parameter values will be specified as RC6-w/r. Of particular relevance to the AES effort will be the versions of RC6 with 16-, 24-, and 32-byte keys. For all variants, RC6-w/r/b operates on units of four w-bit words using the following six basic operations. The base-two logarithm of w will be denoted by lgw.

- a + b integer addition modulo 2^w
- a - b integer subtraction modulo 2^w
- a $\oplus$ b bitwise exclusive-or of w-bit words
- a $\times$ b integer multiplication modulo 2^w

- a<<<b rotate the w-bit word a to the left by the amount

Given by the least significant lgw bits of b

a>>>b rotate the w-bit word to the right by the amount given by the least significant lgw bits of b.

RC6 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 first byte of plaintext or ciphertext is placed in the least-significant byte of A; the last byte of plaintext or ciphertext is placed into the most-significant byte of D. Use the (A;B;C;D) = (B;C;D;A) to mean the parallel assignment of values on the right to registers on the left.

Encryption with RC6 w/r/b
Input: Plaintext stored in four w-bit input registers A;B;C;D
Number r of rounds
w-bit round keys S[0; : : : ; 2r + 3]
Output: Ciphertext stored in A;B;C;D
Procedure:
B = B + S[0]
D = D + S[1]
for i = 1 to r do
begin
t = (B × (2B + 1))<<<lgw
u = (D × (2D + 1))<<<lgw
A = ((A ⊕ t)<<<u) + S[2i]
C = ((C ⊕ u)<<<t) + S[2i+ 1]
(A;B;C;D) = (B;C;D;A)
end
A = A + S[2r + 2]
C = C + S[2r + 3]

Decryption with RC6 w/r/b
Input: Ciphertext stored in four 32-bit input registers A;B;C;D
Number r of rounds
w-bit round keys S[0; : : : ; 2r + 3]
Output: Plaintext stored in A;B;C;D
Procedure:
C = C - S[2r + 3]
A = A- S[2r + 2]
for i = r down to 1 do
begin
(A;B;C;D) = (D; A;B;C)
u = (D × (2D + 1))<<<lgw
t = (B × (2B + 1))<<<lgw
C = ((C - S[2i + 1])>>>t) ⊕u
A = ((A - S[2i])>>>u) ⊕ t
end
D = D - S[1]
B = B - S[0]

4 The SRC Algorithm

Like RC6 and Serpent, SRC is a fully parameterized family of encryption algorithms. Since the SRC algorithm combined between them to take advantage of the powerful supported by RC6 and Serpent algorithms. SRC is a block cipher that encrypts data in 16-byte blocks. The algorithm consists of two parts: a key-expansion part and a data-encryption part. Key expansion will be adopted from Serpent algorithm. SRC has 32 rounds. Each round consists of a key-dependent permutation and substitution. The substitution will be adopted from Serpent and key-dependent permutation from RC6.

4.1 Encryption and Decryption

The underlying philosophy behind SRC is that simplicity of designed yields algorithm that is both easier to implement through the use of the RC6 algorithm instead of linear transformation and a simple S-box substitution. Thus makes up the body of the SRC to be as simple as possible, while still retaining the desirable cryptographic properties of the structure.

Figure (1) illustrates the architecture of the SRC algorithm with 32-rounds. The input plaintext is a 128-

bit data element, and the output is the ciphertext is 128 bits.

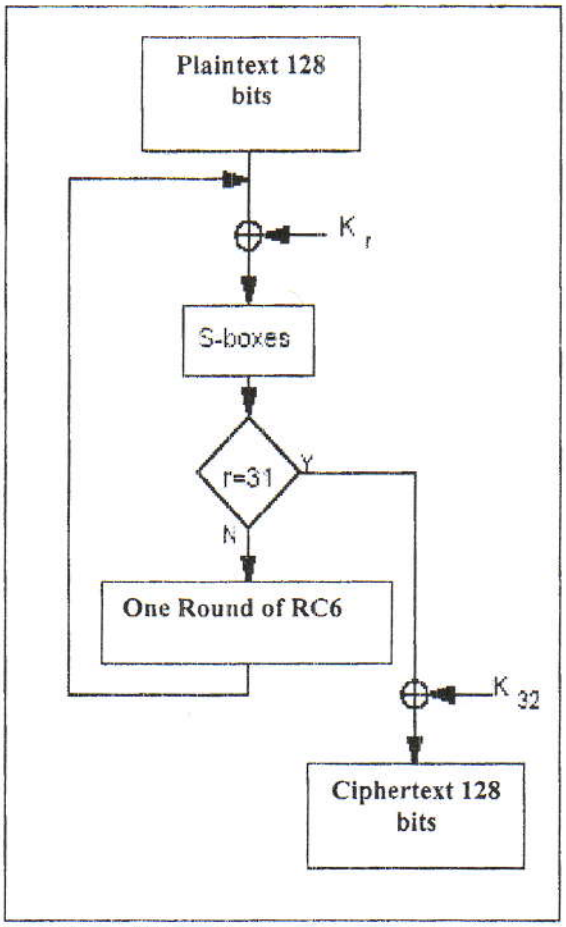

Figure 1 The SRC Algorithm

4.2 Initial and Final Permutation

The first step in the algorithm is the Initial Permutation IP. It is like Serpent [2], which is applied before the first. It is required to provide the necessary diffusion and confusion to the input block. As with Serpent, the Final Permutation FP is the inverse of the initial permutation. The initial and final permutations are used to simplify an optimized implementation of the cipher and to improve its computational efficiency. RC6 did not have reversible mixing function before and after the last round [3]. This would further confuse the entry values into the Feistal network and ensure a complete avalanche effect after the first two rounds.

The SRC algorithm is a SP (Substitution and Permutation) network consisting of 32 iterations of (RC6-function and F-function). Thus the cipher may be formally described by the following algorithm:

The SRC Algorithm

Input: Plaintext 128-bits P.
Output: Ciphertext 128-bits C.
Begin:

$B_0 = IP(P)$

$B_{i+1} = R_i(B_i)$

$C = FP(B_r)$

Where

$R_i(X) = RC6(S_i(X_i \oplus K_i))$ I = 0, …. , r-2

$R_i(X) = S_i(X \oplus K_i) \oplus K_i$ I = r-1

End.

The RC6 algorithm used adding as a reversible mixing function before the first and after the last round. While the SRC algorithm will be used more secure the IP and FP.

Decryption is exactly the same as encryption, except that keys are used in the reverse order. The encryption and decryption algorithms have the same complexity.

4.3 The Function

The RC6 is required to provide the necessary diffusion and confusion to the output of the S-box, such that additive differences will be destroyed as the key is changed. This could provide a protection against linear and differential cryptanalysis (see figure 2) [3].

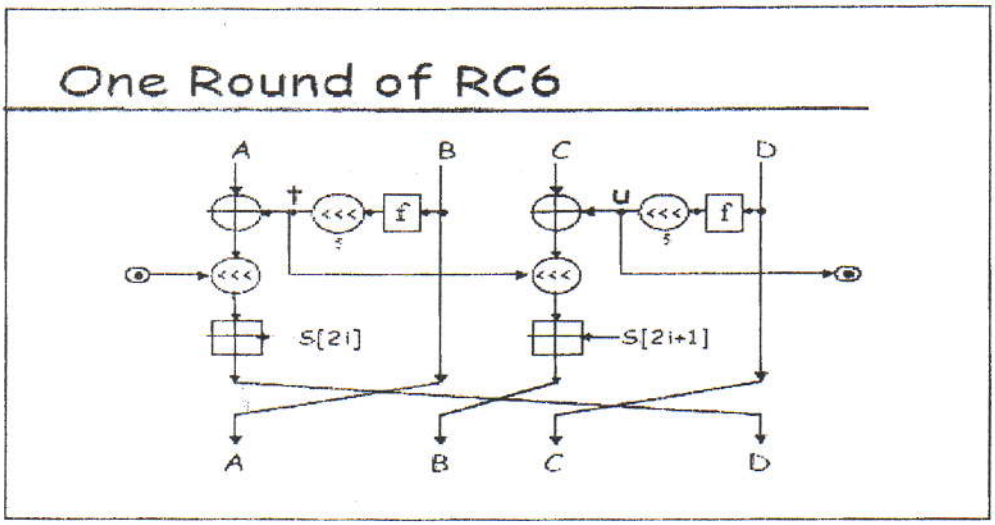

Figure 2 One round of RC6

The previous Serpent algorithm used fixed table to performed diffusion to the output of the S-box which is also used fixed table. While the SRC algorithm used key dependent function this is more secure than fixed table [4].

In Figure (3), F-function has 128-bits input, 128-bit input subkey K_i and 128-bit output Z and it adopts SP (Substitution-Permutation network) structure. The non-reversible function is used for strength, speed, and simplicity. The function combines the S-box outputs is as fast as possible.

However, until recently multiplication were considered prohibitively expensive for fast encryption. This was true since old machines took many cycles to perform a single multiplication operation. Today, this is no longer the case, as essentially all modern architectures (including PowerPC, Pentium-Pro, Alpha, Ultra-SPARC, and others) support a multiply instruction which takes about two cycles to complete. The usage of multiplication in RC6 has to do with our ability to analyze them: Analyzing a multiplication of two data words turns out to be a very hard task.

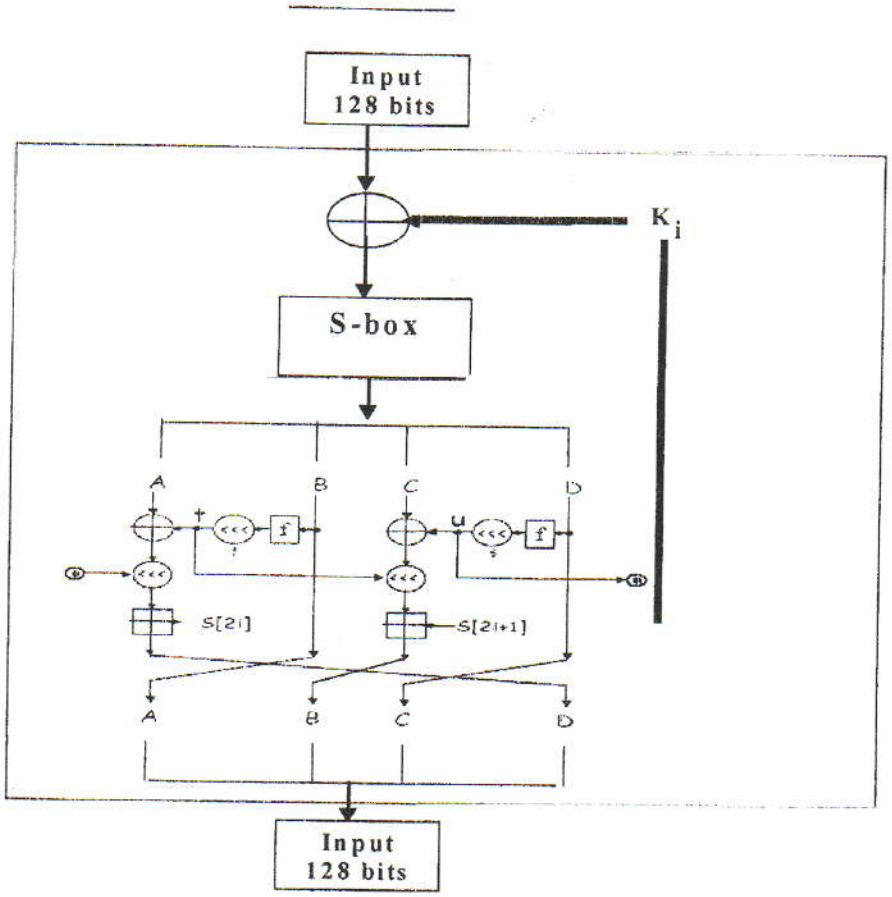

Figure 3 The F- Function the SRC Algorithm

5. Security of SRC Algorithm

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

The SRC algorithm adopts RC6 and Serpent algorithm to meet the AES requirement and overcomes the weaknesses of these algorithms by the following:

1. Speed up algorithm by using one round of RC6 instead of Linear Transformation (fixed table) which is used by previous algorithm.
2. SRC increased the complexity of algorithm by mixing operation from different algebraic group. XOR, addition, rotation and multiplication.
3. SRC adopts key-dependent permutations to provide protection against differential and linear cryptanalysis.
4. Each function of the proposed algorithm is dependent on its key, so this will prevent fixed output and increase the non-linearity of the algorithm.

5.1 Differential and Linear Cryptanalystsis

Differential and linear cryptanalystsis are worked against block cipher algorithms that use constant S-boxes. These attacks are heavily dependent on the structure of S-boxes [4]. The SRC algorithm is patient to these types of attacks and this belongs to many reasons:

- These types of attacks are 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 is used and this could provide protection against linear and differential cryptanalysis.

5.2 Measuring the Complexity

The strength of a cipher is determined by the computational complexity of the algorithms used to solve the cipher. The strength of an algorithm is measured by its time (T) and space (S) requirements. The time and space requirements grow, as the size of input is increases.

The evaluation of improved algorithm can be done through analyzing the computing time, which is performed by studying the frequency of execution of its statements given various sets of data. The common notion for evaluating an algorithm is the concept of the order of magnitude of the time complexity and its expression by sympototic notation. If T (n) is the time for an algorithm on n inputs, then it is written [6]:

$$T(n) = \theta(f(n))$$

In this section the complexity of one round from cryptanalysis of the view to the two algorithms Serpent and improved algorithm are computed such as following:

5.2.1 The complexity of Serpent algorithm

Each round of Serpent uses one EX-oring of 128-bits and Linear Transformation (LT). The complexity is as follows:

$$\theta(n(2^7)) * \theta(n(2^3)) * \theta(n(2^7)) * \theta(n(2^3)) * \theta(2^{4n-1}))$$

$$\theta(\log[\, n\,2^7 + n\,2^3 + n\,2^7 + n\,2^3 + 2^{4n-1}\,])$$

$$\theta(\log[272\,n + 2^{4n-1}]$$

$$\theta(e^{272n + 2^{4n-1}})$$

$$\equiv \theta(e^{4n}) \quad \text{where } n=32$$

5.2.2 The complexity of the SRC algorithm

The complexity of one round of the SRC algorithm is equivalent to the complexity of one round of Serpent except LT and one round of RC6 algorithm:
The complexity is as follows:

- **The complexity of one round of RC6:**

$$\theta(4(2^{4n-1})) * \theta(4(n/4)) * \theta(2(2^{4n+1})))$$

$$\theta(\log[\, 4(2^{4n-1}) + 4(n/2) + 2(2^{4n+1})\,])$$

$$\theta(\log[\, 2^{16n-2} + 4(n/2) + 2^{8n+2}\,])$$

$$\theta(\log[\, 2^{24n}\,])$$

$$\equiv \theta(e^{24n}) \quad \text{where } n=32$$

- **The total complexity of improved SRC is:**

$$\theta(\log[2^{4n-1} + 2^{24n}\,]$$

$$\equiv \theta(e^{28n})$$

5.3 Plaintext-Ciphertext Independence

This test is used to test the independence between the plaintext and ciphertext by using the following steps:

1. Generate large number of random plaintext Pr where (r=1...R) and R is the number of plaintext which will be tested.
2. Encrypt the chosen plaintext by using key K to obtain the corresponding ciphertext Cr.
3. generate a new blocks which are called Independent Blocks (INr) by using the following equation:

INr=Pr ⊕ Cr r=1..R

The algorithm to generate the independent blocks

```
F1=plaintext.txt
F2=ciphertext.txt
For i= 1 to R do
  For j = 1 to n do
     Read(f1,ch1)
     Read(f2,ch2)
     If ch1=ch2
     Then
        IN-ARY[I,j]='0'
     Else
        IN-ARY[I,j]='1'
     End for
End for
```

Where n is the block length and R is the number of blocks.

The independent blocks must be random to assign that the ciphertext independent to plaintext [5]. The following randomness test will be used to test these blocks:

The Frequency test computes the frequencies to check the randomness of the independent blocks. Suppose the ***ai*** is the actual number of the block which is having the Hamming weight ***i*** from the **R** independent blocks. The *ei* which represents the expected number of the block which is having the Hamming weight ***i*** from the **R** independent blocks.

$$e_i = \frac{\binom{n}{i} \times R}{2^n} \quad i=1...n$$

The algorithm computes actual frequencies of independent blocks a_i is as follows:

```
   For I = I to R do
   ai=0
   For j = 1 to n do
If IN-ARY[I,J]='1'
Then  ai=ai+1;
End for
ai_ary[i]=ai
End for
```

A file contained 128 random blocks will be encrypted by using Serpent, RC6 and the SRC algorithm. After that the above frequency test are used. The Figure 4 show results that will be obtained.

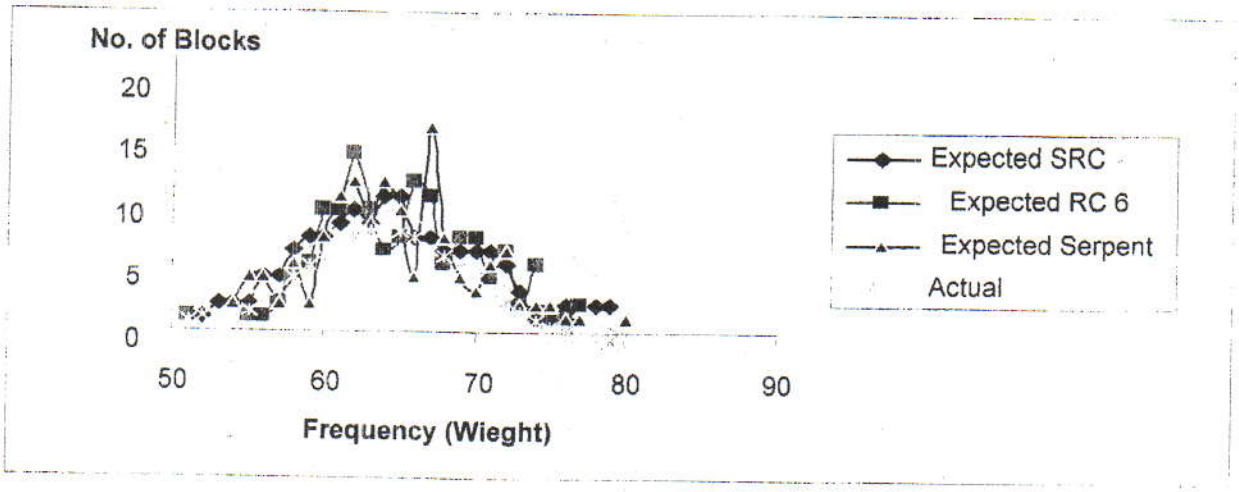

Figure 4 Frequency Test

Figure 4 shows that the expected frequencies of independent blocks of the improved SRC algorithm are approximated to the actual frequencies of independent. While the Serpent and RC6 algorithms are not approximated. The reason for these results is that the expected numbers for all independent blocks are less than actual numbers when implements the improved SRC algorithm. This means that the improved SRC algorithm is the best one between them.

5.4 Avalanche Effect

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

1. **ffffffffffffffffffffffffffffffff**
2. **aaaaaaaaaaaaaaaaaaaaaa**
3. **ccccccccccccccccccccccccc**
4. **22222222222222222222**
5. **ddddddddddddddddddddddd**
6. **4444444444444444444444**

And using a key of length 32 bytes where the key is "**30000000000000000000000000000000**".

Horst Feistel referred to the avalanche effect as: "a small change in the key gives rise to a large change in the ciphertext" [5]. Table 1 show the avalanche effect on the plaintext when only one bit is changed in the key by using a Serpent algorithm.

Table 1 Avalanche Effect of Serpent Algorithm: Change one bit in key

Block No.	Ciphertext 128-bits in Hexadecimal	Avalanche
1	8950b3b67ef833522b687c04ed44ce88 8d0ca473e0e74467523b0e8c91a3e39	65
2	b07b73a9cd7f87fee00e63aa8a3d583b f57971fb93c5fe51fe56c0391585a991	63
3	95a2fdb5794dbbbd8fa8da6d9b458b4f ab3ae8d0ee0ca37aaf8e484d776c303a	56
4	85ad5a03179b16aa973974e72fff519a d43e4326182d904682baa8d23ff2e43c	58
5	5a3db61011900cdbeddd39cd8f89801 d66a20d33805426852e1ac062b04f918	65
6	f5ab3d9e66893748c5a7ce3cde23961c fad88143e7a1364d61e627281980033d	54
Key1	30000000000000000000000000000000	
Key2	20000000000000000000000000000000	

Table 2 shows the avalanche effect on the plaintext when only one bit is changed in the key by using the SRC algorithm.

Table 2 Avalanche Effect of SRC Algorithm: Change one bit in key

Block No.	Ciphertext 128-bits in Hexadecimal	Avalanche
1	1c54ade9fb1a973c4c15867cf2cb0eb9 8f8176d5edddf86cc5c78063c5483dcc	66
2	c2899b507b42c78a4f845c4b915b2061 3d3afa37a7acfadbd0d6d027f6eac5e9	72
3	d075284af00a150a7f85750a9c60020 e8cb69b2ea54c03f85db46c562cf939	70
4	7bc1377ce16eeac569ca12c1044335e 2d9cb5accffb750b8348a0746ffb6a43	76
5	a3721daf70bb9b3740e68e14b32cc5d4 a140606bfeaf74651d3308faa4906b1	68
6	6f6221d4634112f6c4423ae0709cd2cf 49c596105dbcec6fe37b74d1b1733948	75
Key1	30000000000000000000000000000000	
Key2	20000000000000000000000000000000	

Table 3 shows the avalanche effect on the plaintext when only one bit is changed in the key by using the RC6 algorithm.

Table 3 Avalanche Effect of RC6 Algorithm: Change one bit in key

Block No.	Ciphertext 128-bits in Hexadecimal	Avalanche
1	1c54ade9fb1a973c4c15867cf2cb0eb9 8f8176d5edddf86cc5c78063c5483dcc	69
2	c2899b507b42c78a4f845c4b915b2061 3d3afa37a7acfadbd0d6d027f6eac5e9	63
3	d075284af00a150a7f85750a9c60020 e8cb69b2ea54c03f85db46c562cf939	59
4	7bc1377ce16eeac569ca12c1044335e 2d9cb5accffb750b8348a0746ffb6a43	67
5	a3721daf70bb9b3740e68e14b32cc5d4 a140606bfeaf74651d3308faa4906b1	68
6	6f6221d4634112f6c4423ae0709cd2cf 49c596105dbcec6fe37b74d1b1733948	57
Key1	30000000000000000000000000000000	
Key2	20000000000000000000000000000000	

Table (1) shows that the average of avalanche effect of Serpent algorithm is 60%. While table (2) shows that the average of the avalanche effect of RC6 algorithm is 63%. But table (3) shows that the average of the avalanche effect of the SRC improved algorithm is 71%. Where the number of block that have weights greater than half of the block length in SRC algorithm is 6 over 6 blocks, 4 over 6 in pervious Serpent algorithm and 3 over 6 in RC6 algorithm.

5.5 Time requirement

In this section the time requirements are computed for the Serpent algorithm, RC6 and the SRC algorithm. Figure (5) shows this test.

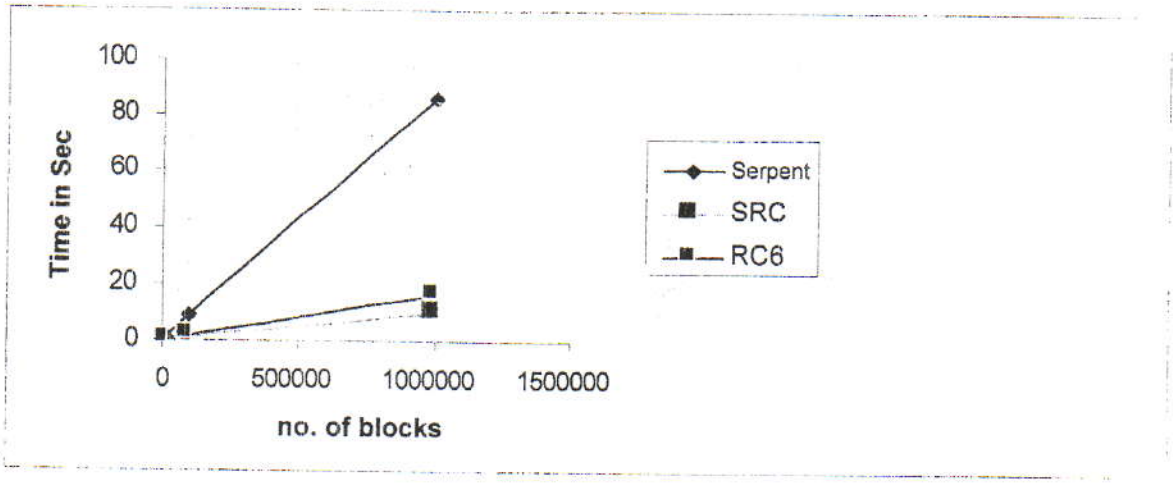

Figure 5 Time comparison between SRC, Serpent, and RC6 algorithms

6. Conclusion

SRC is a secure, compact and simple block cipher. It offers good performance a considerable exibility. Furthermore its simplicity will allow analysts to quickly refine and improve our estimates of its security.

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

The SRC algorithm uses a new a complicated F-function which is combined between permutations and substitutions (SP network). This will be making a strong and secure cipher. On the other hand the SRC algorithm mixed operation from different algebraic group: XOR, addition, rotation and multiplication. The SRC uses the same algorithm for encryption and decryption with the same key schedules, and supports variable key-length up to 256 bytes. The encryption algorithm and key schedule must be designed in tandem; subtle changes in one affect the other. It is not enough to design a strong round function and then to graft a strong key schedule onto it (unless you are satisfied with an inefficient and inelegant construction); both must work together.

From the results that were obtained in section 5 and after measuring the strength of the proposed algorithm. The concluded is that this algorithm has average of the avalanche effect of the SRC algorithm is 71% while the average of the avalanche effect for Serpent and RC6 is 60%, 63% respectively. After measuring the complexity we obtained that the complexity of the SRC algorithm is $\theta(e^{28n})$ while the complexity of RC6 is $\theta(e^{24n})$ and the complexity of Serpent is $\theta(e^{4n})$. This will make it very hard to mount any statistical cryptanalytical attacks compared with RC6 and Serpent algorithms.

The SRC algorithm is fastest one among them because of using one round of RC6 algorithm instead of Linear Transformation in each round in previous Serpent algorithm as it is clear from figure (5).

Refrences

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. R. Anderson, E. Biham, and L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard, " First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.

3. R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, "The RC6TM Block Cipher," First Advanced Encryption Standard (AES) Conference, Ventura, CA, 1998.

4. Bruce Shiner "Applied Cryptography Second Edition Protocols. Algorithms, and Source, and Source Code in C", John Wiley and Sons, Inc., 1996.

5. Khalid S. Hameed. " The statistic tests of feedback symmetric block cipher system", M. Sc. , Thesis Univ. of tech, Baghdad, 1997.

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

7. 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.