Proposal of New Keys Generator for DES Algorithms Depending on Multi Techniques

Dr. Alaa kadhim
Applied Sciences of Department, University of Technology/ Baghdad
Email: Dralaa81.uotechnology.edu.iq
Mohammed salih
Applied Sciences of Department, University of Technology/ Baghdad

Received on: 6/3/2013 & Accepted on: 4/7/2013

ABSTRACT

The requirements of an ordinary PRNG are also satisfied by a cryptographically secure PRNG, but the reverse is not true. CSPRNG requirements fall into two groups: first, that they pass statistical randomness tests; and secondly, that they hold up well under serious attack, even when part of their initial or running state becomes available to an attacker. in this paper, design and implementation new keys generator depend on multi of techniques as (logic circuit XOR, NOT and AND) in one proposal also in another proposal used the coding, permutations and reorder bit by search method in artificial intelligent also depend of polynomial equations to expand the original key, the generator product random runs of bits after check by statistical test.

Keywords: Cryptography, Keys Generator, Randomness, Symmetric Key, Lfsrs

أقتراح مولد مفاتيح جديد لخوارزميات التشفير القياسي بالاعتماد على تعدد التقنيات

الخلاصه

ان من المتطلبات الاساسيه هو تكوين مولد ارقام عشوائيه أمن لكي يكون النظام قوي ولكن لايتكون لدينا نظام تشفير أمن بعدم وجود مولد مفاتيح قوي ويمكن اختبار ان المولد قوي يجب ان يتجاوز الاختبارات العشوائيه الخاصه بالمواصفات الاحصائيه وكذلك ان يكون غير قابل للكسر من خلال سلسله من الهجمات على النصوص المشفره ان في هذا البحث تم تصميم وتنفيذ مولد مفاتيح عشوائيه لخوارزميات التشفير القياسي بالاعتماد على تعدد التقنيات حيت استخدمت في المولد الاول البوابات المنطقيه اما في المولد الثاني استخدمت نظريه الترميز وكذلك طرق تغير الترتيب بالاعتماد على طرق البحث في الذكاء الاصطناعي وتم اجراء الاختبارات على المفاتيح الناتجه للتحقيق العشوائيه .

INTRODUCTION

Historically, the term "cryptography" has been associated with the problem of designing and analysing encryption schemes (i.e., schemes that provide secret communication over insecure communication media) [1]. Cryptography is the art of secret writing. It involves transforming information into apparently unintelligible garbage so that unwanted eyes will be unable to comprehend it [2]. To be involved with modern cryptography is to dive willy-nilly into number theory, that is, the study of the natural numbers, one of the most beautiful areas of mathematics., there is no limit to the depth of involvement of number theory with cryptography, and many significant mathematicians have made important contributions to this area [3]. Security has many facets. For a system to be secure, many factors must combined. For example, it should not be possible for hackers to exploit bugs, break into a system, and use an account. They shouldn't be able to buy off your system administrator. They shouldn't be able to steal your back-up tapes. These things lie in the realm of system security. The cryptographic protocol is just one piece of the puzzle. If it is poorly designed, the attacker will exploit that. For example, suppose the protocol transmits a password in the clear (that is, in a way that anyone watching can understand what it is), that is a protocol problem, not a system problem. In addition, it will certainly be exploited [4].

FEISTEL MODE

Many block ciphers have a Feistel structure. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves. Where these terms are defined as follows:

- Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding cipher text element or group of elements.
- Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed [5].

BASIC REQUIREMENTS OF BLOCK CIPHER

In fact, Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions, Where these terms are defined as follows:

- Diffusion: the statistical structure of the plaintext is dissipated into long-range statistics of the cipher text. This is achieved by having each plaintext digit affect the value of many cipher text digits; generally, this is equivalent to having each cipher text digit be affected by many plaintext digits.
- Confusion: seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the cipher text, the way in which the key was used to produce that cipher text is so complex as to make it

difficult to deduce the key. This is achieved by the use of a complex substitution algorithm [5].

DATA ENCRYPTION STANDARD (DES)

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), For DES; data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.[5].The encryption transformation depends on a 56-bit secret key and consists of sixteen Feistel iterations surrounded by two permutation layers: an initial bit permutation IP at the input, and its inverse IP−1 at the output. The structure of the cipher is depicted in Figure (1). The decryption process is the same as the encryption, except for the order of the round keys used in the Feistel iterations.

The 16-round Feistel network, which constitutes the cryptographic core of DES, splits the 64- bit data blocks into two 32-bit words, L Block and R Block (denoted by L_0 and R_0). In each iteration (or round), the second word Ri is fed to a function f and the result is added to the first word Li. Then both words are swapped and the algorithm proceeds to the next iteration. The function f is key-dependent and consists of four stages:

1. Expansion (E). The 32-bit input word is first expanded to 48 bits by duplicating and reordering half of the bits.
2. Key mixing. The expanded word is XORed with a round key constructed by selecting 48 bits from the 56-bit secret key; a different selection is used in each round.
3. Substitution. The 48-bit result is split into eight 6-bit words which are substituted in eight parallel 6 × 4-bit S-boxes. All eight S-boxes are different but have the same special structure.
4. Permutation (P). The resulting 32 bits are reordered according to a fixed permutation before being sent to the output.

The modified R Block is then XORED with L Block and the resultant fed to the next R Block register. The unmodified R Block is fed to the next L Block register. [6].

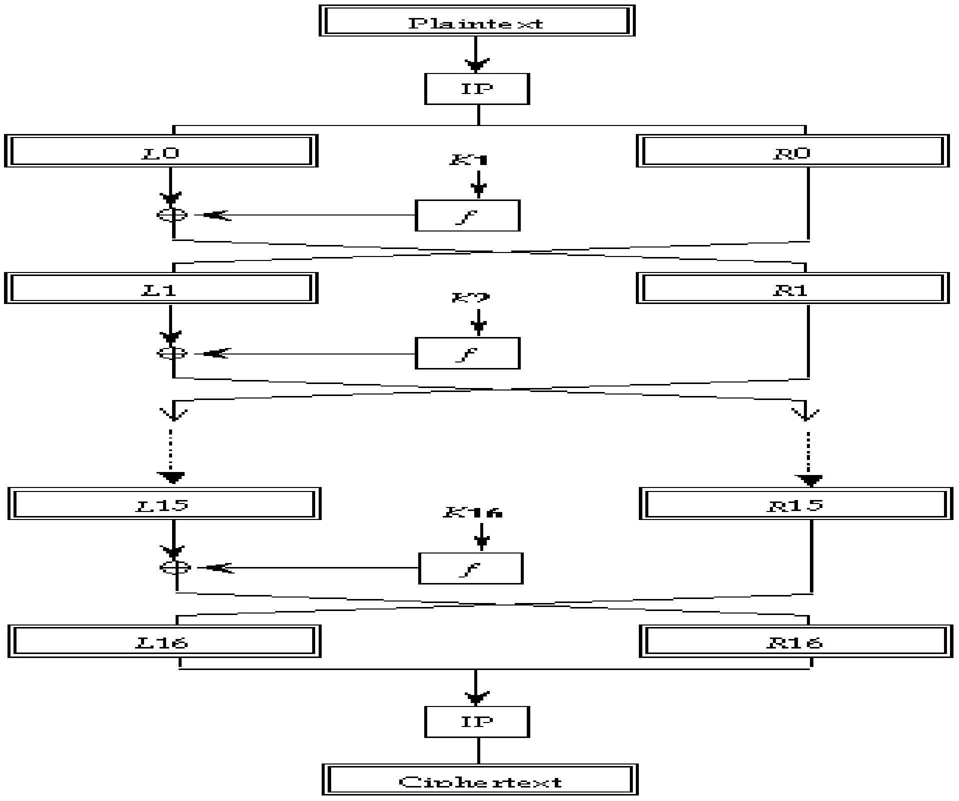


Figure (1) DES computation path.

FIVE BASIC TESTS

Let s = s0; s1; s2; sn−1 be a binary sequence of length n. This subsection presents five statistical tests that are commonly used for determining whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to exhibit. It is emphasized again that the outcome of each test is not definite, but rather probabilistic. If a sequence passes all five tests, there is no guarantee that it was indeed produced by a random bit generator.

(i) **Frequency test (monobit test)**

The purpose of this test is to determine whether the number of 0's and 1's in s are approximately the same, as would be expected for a random sequence. Let n0, n1 denote the number of 0's and 1's in s, respectively. The statistic used is

$$X_1 = (n0 - n1)^2 /n \qquad \dots (1)$$

Which approximately follows a α^2 distribution with 1 degree of freedom if $n \geq 10$.

(ii) **Serial test (two-bit test)**

The purpose of this test is to determine whether the number of occurrences of 00, 01, 10, and 11 as subsequences of s are approximately the same, as would be expected for a random sequence. Let n0, n1 denote the number of 0's and 1's in s, respectively, and let n00, n01, n10, n11 denote the number of occurrences of 00, 01, 10, 11 in s, respectively. Note that n00 + n01 + n10 + n11 = (n − 1) since the subsequence's are allowed to overlap. The statistic used is

$$X_2 = 4/n - 1\,(n^2_{00} + n^2_{01} + n^2_{10} + n^2_{11}) - 2/n\,(n^2_0 + n^2_1) + 1 \qquad \ldots (2)$$

Which approximately follows a α^2 distribution with 2 degrees of freedom if $n \geq 21$.

(iii) **Poker test**

Let m be a positive integer such that $[n/m] \geq 5.(2^m)$, and let $k = [n/m]$. Divide the sequence s into k non-overlapping parts each of length m, and let n_i be the number of occurrences of the ith type of sequence of length m, $1 \leq i \leq 2^m$. The poker test determines whether the sequences of length m each appear approximately the same number of times in s, as would be expected for a random sequence. The statistic used is

$$X_3 = 2^m/k\ \left(\sum_{i=1}^{2m} n_i^2\right) - k \qquad \ldots (3)$$

Which approximately follows a α^2 distribution with $2^m - 1$ degrees of freedom. Note that the poker test is a generalization of the frequency test: setting m = 1in the poker test yields the frequency test.

(iv) **Runs test**

The purpose of the runs test is to determine whether the number of runs (of either zeros or ones) of various lengths in the sequence s is as expected for a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length n is $e_i = (n-i+3)/2^{i+2}$. Let k be equal to the largest integer i for which $e_i \geq 5$. Let B_i, G_i be the number of blocks and gaps, respectively, of length i in s for each i, $1 \leq i \leq k$. The statistic used is

$$X_4 = \sum_{i=1}^{K} \frac{(b_i - e_i)^2}{e_i} + \sum_{i=1}^{K} \frac{(g_i - e_i)^2}{e_i} \qquad \ldots (4)$$

Which approximately follows a α2 distribution with 2k − 2 degrees of freedom

(v) **Autocorrelation test**

The purpose of this test is to check for correlations between the sequence s and (non-cyclic) shifted versions of it. Let d be a fixed integer, $1 \leq d \leq [n/2]$. The number of bits in s not equal to their d-shifts is

$A(d) = \sum_{i=0}^{n-d-1}$ s_i_+s_{i+d}, where + denotes the XOR operator. The statistic used is

$$X_5 = 2(A(d) - n - d/2)/\sqrt{n-d} \qquad \ldots (5)$$

Which approximately follows an N (0, 1) distribution if $n - d \geq 10$. Since small values of A (d) are as unexpected as large values of A (d), a two-sided test should be used [7, 8].

ARCHITECTURE PROPOSAL DYNAMIC KEYS GENERATOR

There are many traditional methods to generate random numbers for use in cryptographic keys. The most famous methods as widely known are Conventional random number generators available in most programming environments but are not suitable for use in cryptographic applications. The point is that the data must be unpredictable for any external observer.

In this paper designed and implementation two proposal dynamic key generators system depended on multi techniques as polynomials and permutation equation to produce random binary bits as secret keys used in DES block cipher.

First Proposal

In this's proposal designed system consist three parts (A, B, C):
The first part (A) have three linear feedback shift registers (LFSR), denoted by AR1, AR2, AR3 with different length 5, 4, 3 respectively with Parallel substructure shifting INPUT - linear feedback shift registers (PSSI-LFSR) as shown in the following Figure.

In order for a particular LFSR to be a maximal-period LFSR, the polynomial formed from a tap sequence plus the constant 1 must be a primitive polynomial. The degree of the polynomial is the length of the shift register. A primitive polynomial of degree n is an irreducible polynomial that divides $x^{2n-1} + 1$, but not XD + 1 for any d that divides $2^n - 1$. In addition, the primitive feedback polynomials choose in part (A) are:

$$AR1=1+x^2+x^5 \quad \dots (6)$$

$$AR2=1+x^1+x^4 \quad \dots (7)$$

$$AR3=1+x^1+x^3 \quad \dots (8)$$

The second part (B) have three linear feedback shift registers (LFSR), denoted by BR1, BR2, BR3 with different length 5,4,3 respectively with Parallel substructure shifting INPUT - linear feedback shift registers (PSSI-LFSR) as shown in following figure In addition, the primitive feedback polynomials choose in part (B) are:

$$BR1=1+x^3+x^5 \quad \dots (9)$$

$$BR2=1+x^3+x^4 \quad \dots (10)$$

$$BR3=1+x^2+x^3 \quad \dots (11)$$

The third part (C) have logic gates (XOR, AND, NOT) connected with the outputs of part (A, B) to produce random key for any block cipher.

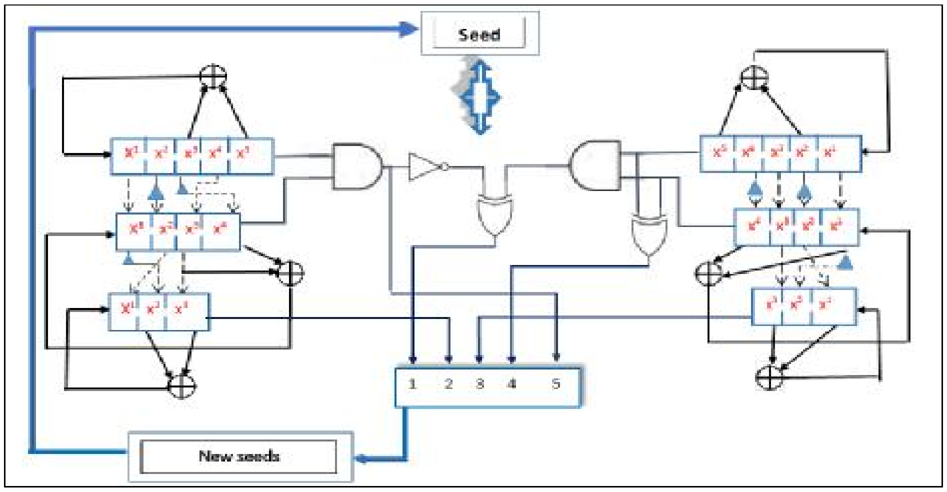


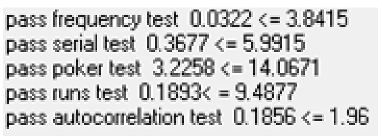
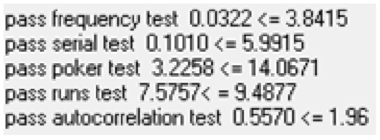
Figure (2) First Proposal.

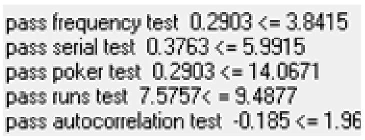
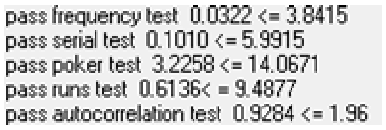
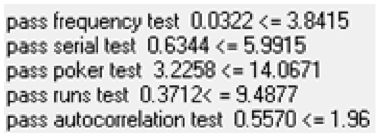
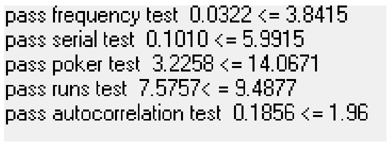
First Proposal Algorithm

INPUT:
Different Initial Seeds to LFSRs of parts **A** and **B**.
OUTPUT:
Random sequence of **(0, 1)**
Process:
Begin://For part **A** For i = 1 to max period do
Take outputs **AR1, AR2 and AR3** respectively and store in three array **A1, A2, and A3** respectively Make Parallel substructure shifting input **AR2 (1) = AR1 (1), AR2 (2) = NOT (AR1 (2)), AR2 (3) = AR1 (3), AR2 (4) = NOT (AR1 (4)), AR3 (1) = AR2 (2), AR3 (2) = flip (AR2 (1)), AR3 (3) = AR2 (3)** Make shift to AR1, AR2, and AR3 //End part **A** //For part **B** For j = 1 to max period do Take outputs **BR1, BR2 and BR3** respectively and store in three array **B1, B2, and B3** respectively Make Parallel substructure shifting input **BR2 (1) = BR1 (1), BR2 (2) = NOT (BR1 (2)), BR2 (3) = BR1 (4), BR2 (4) = NOT (BR1 (3)), BR3 (1) = BR2 (2), BR3 (2) = NOT (BR2 (1)), BR3 (3) = BR2 (3)** Make shift to BR1, BR2, and BR3 //End part **B** Make merge to arrays register systems to represent random key for any block cipher in the fallowing form: (A1 And A2) Xor (NOT ((B1 And B2))) **concat** B3 **concat** A3 **concat** A1 Xor A2 **concat** B3 **concat** (B1 And B2) Selected bit from result of merge arrays to present new seeds for parts A, B. **End**

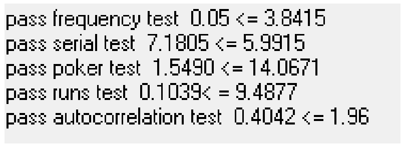
First proposal Experiment and results

Initial Seeds to LFSR of part A
AR1: 11100, AR2: 1111, AR3: 001
Initial Seeds to LFSR of system B
BR1: 00101, BR2: 0110, BR3: 111
After programing and check the output for result parts A, B are randomize:

 pass frequency test 0.0322 <= 3.8415 pass serial test 0.3677 <= 5.9915 pass poker test 3.2258 <= 14.0671 pass runs test 0.1893< = 9.4877 pass autocorrelation test 0.1856 <= 1.96 AR1	 pass frequency test 0.0322 <= 3.8415 pass serial test 0.1010 <= 5.9915 pass poker test 3.2258 <= 14.0671 pass runs test 7.5757< = 9.4877 pass autocorrelation test 0.5570 <= 1.96 BR1

 pass frequency test 0.2903 <= 3.8415 pass serial test 0.3763 <= 5.9915 pass poker test 0.2903 <= 14.0671 pass runs test 7.5757< = 9.4877 pass autocorrelation test -0.185 <= 1.96 AR2	 pass frequency test 0.0322 <= 3.8415 pass serial test 0.1010 <= 5.9915 pass poker test 3.2258 <= 14.0671 pass runs test 0.6136< = 9.4877 pass autocorrelation test 0.9284 <= 1.96 BR2
 pass frequency test 0.0322 <= 3.8415 pass serial test 0.6344 <= 5.9915 pass poker test 3.2258 <= 14.0671 pass runs test 0.3712< = 9.4877 pass autocorrelation test 0.5570 <= 1.96 AR3	 pass frequency test 0.0322 <= 3.8415 pass serial test 0.1010 <= 5.9915 pass poker test 3.2258 <= 14.0671 pass runs test 7.5757< = 9.4877 pass autocorrelation test 0.1856 <= 1.96 BR3

The final output sequence for the system check by the test measure randomization and the results:



```
pass frequency test  0.05 <= 3.8415
pass serial test  7.1805 <= 5.9915
pass poker test  1.5490 <= 14.0671
pass runs test  0.1039< = 9.4877
pass autocorrelation test  0.4042 <= 1.96
```

Second Proposal

In this proposal have been proposed random generator has the ability to generate a sequences of (0, 1) characterized by high randomly by relying on a set of technologies used to configure the system explained in the following stages as show in Figure (6).

Expand

This stage relies on the basic values (8-bit) that entering by User then expending by depending on primitive polynomial with length (8) to generate (256 bit) randomly in the fallowing form:

$$1+x^3+x^5+x^7+x^8 \qquad \dots (12)$$

Split

At this stage the output of the first stage (256 bit) was divided into four groups (each group consist 64 bits) Where these groups are defined as follows:

Tree search

At this stage, used Search techniques of Artificial Intelligence to spreading 64 bits of first group using depth first search and collecting bits by using breadth first search AI techniques, thus lead to waste every statistical standards then store result as show in Figure (3).

Input tree indexing:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64

Output tree indexing:

1 33 2 18 34 49 3 11 19 26 35 42 50 57 4 8 12 15 20 23 27 30 36 39 43 46 51 55 58 61 5 6 7 8 9 10 13 14 16 17 21 22 24 25 28 29 31 32 37 38 40 41 44 45 47 48 52 53 56 57 59 60 62 63 64

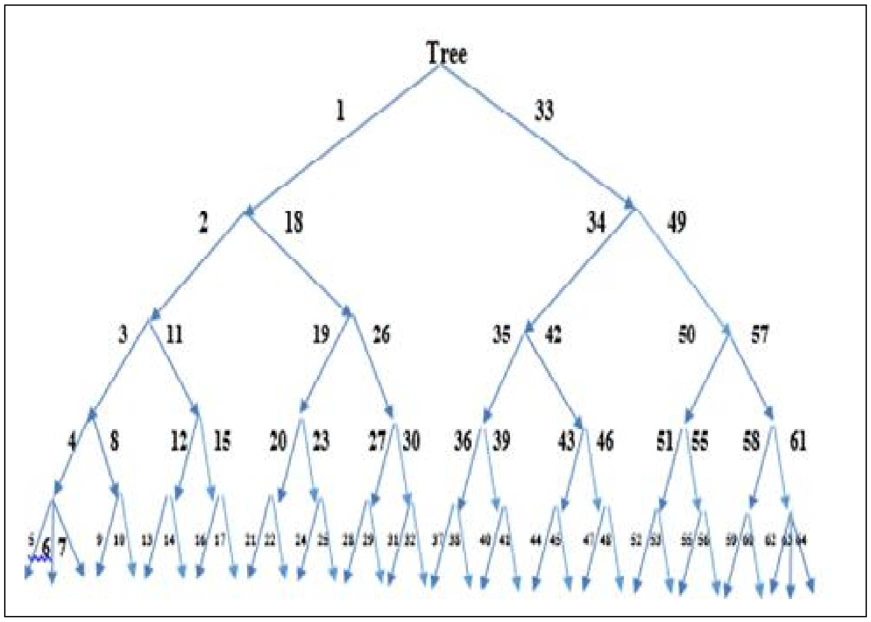


Figure (3) tree search.

Zigzag

At this stage 64 bits of group 2 split two subgroup (each group consist of 32 bits).start operation by taking odd index of second subgroup and even index of first subgroup then taking odd index of first subgroup and even index of second subgroup. As show in Figure (4).

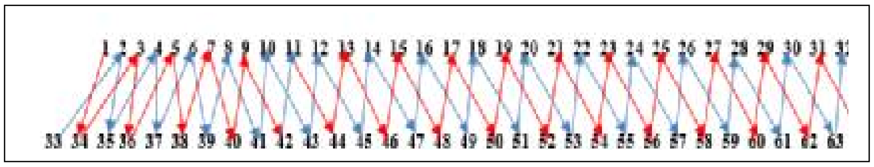


Figure (4) Zigzag.

Output zigzag indexing:
33 2 35 4 37 6 39 8 41 10 43 12 45 14 47 16 49 18 51 20 53 22 55 24 57 26 59 28 61 30 63 32 1 34 3 36 5 38 7 40 9 42 11 44 13 46 15 48 17 50 19 52 21 54 23 56 25 58 27 60 29 62 31 64.

Swap coding

At this stage 64 bits of group 3 split two subgroup (each group consist of 32 bits). Make coding on two sub groups (each group consist of eight characters that's mean one character for each four bits) can be described in the following table form:

Table (1) Swap coding.

0000 = A	0001 = B	0010 = C	0011 = D
0100 = E	0101 = F	0110 = G	0111 = H
1000 = I	1001 = J	1010 = K	1011 = L
1100 =M	1101 = N	1110 = O	1111 = P

Make swap between subgroups coding, where ccc1 presented the first sub group, ccc2 presented second sub group. Swap as follow:

ccc1 (2) $\longleftrightarrow$ ccc2 (3)
ccc1 (3) $\longleftrightarrow$ ccc2 (2)
ccc1 (6) $\longleftrightarrow$ ccc2 (7)
ccc1 (7) $\longleftrightarrow$ ccc2 (6)

Make decoding by reverse method that's use in coding described in the Table.Then make Xor between two sub groups two to produce 32 bits as show in Figure (5).

Table (2) Swap decoding.

A=0000	B=0001	C=0010	D=0011
E=0100	F=0101	G=0110	H=0111
I=1000	J=1001	K=1010	L=1011
M=1100	N=1101	O=1110	P=1111

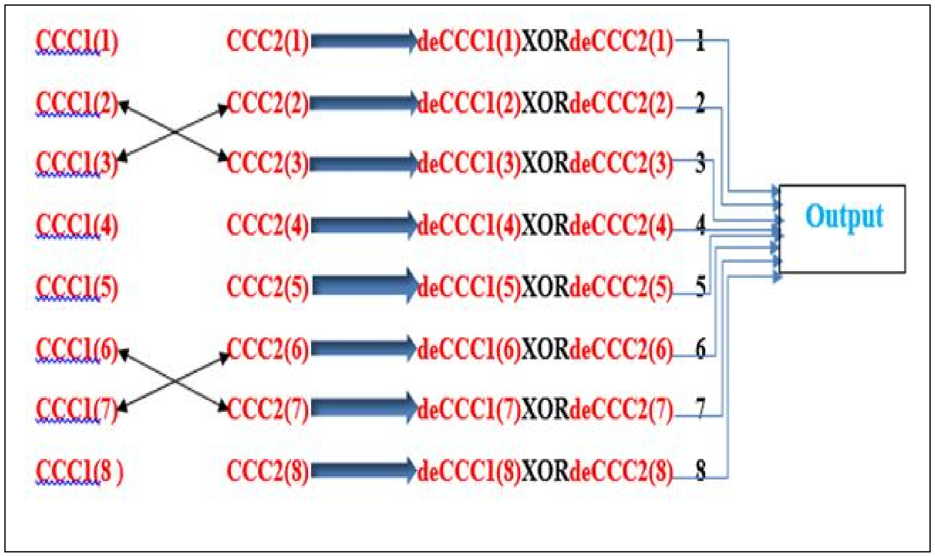


Figure (5) Swap coding.

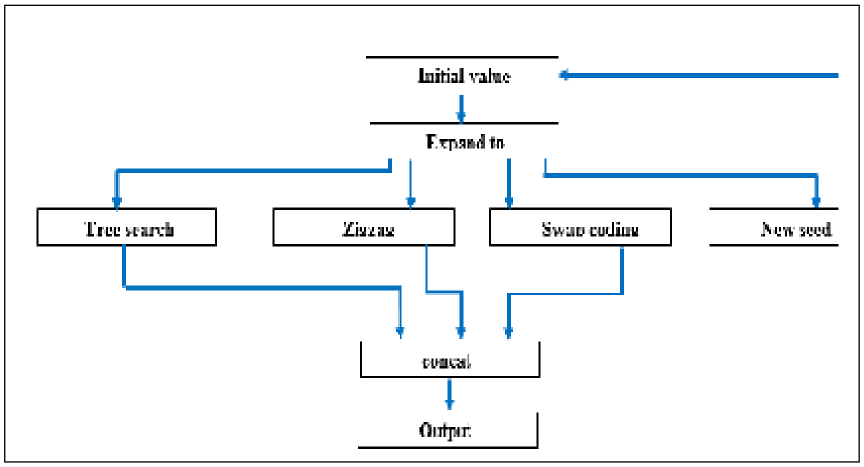


Figure (6) Proposal 2.

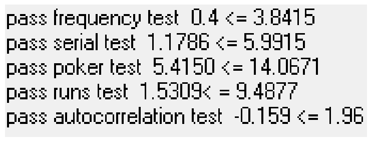
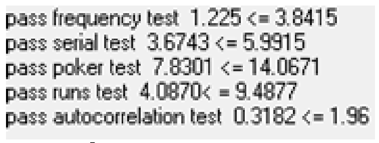
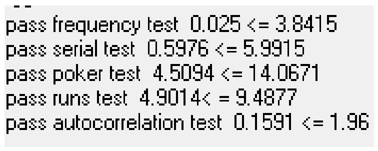
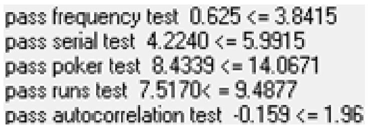
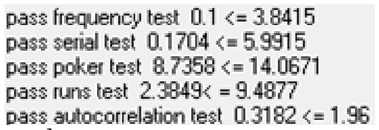
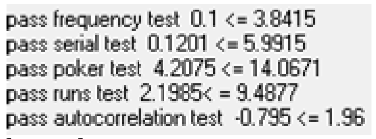
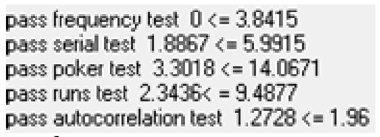
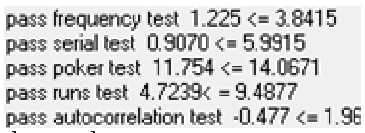
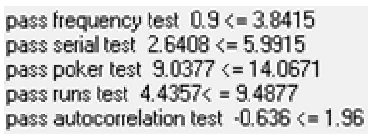
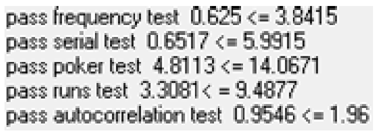
Second Proposal Algorithm

```
Input:  Initial Seed (8) bits
Output: Random sequence of (0, 1) for any block cipher
Process:
Begin:
 Expand initial seed
 Split to four groups
//For group 1 (tree search)
   Spreading bits by using depth first search and collecting bits by using breadth first search
//For group 2 (zigzag)
    Make zigzag on two sub groups
 //For group 3 (swap coding)
    Make coding on two sub groups
    Make swap between subgroups coding
    Make decoding
    Make Xor between two sub groups
    Concat the result of groups 1, 2, 3 respectively to provide random key for any block cipher
//For group4 (new seed)
k = 1
For j = 1 To 8
seed(j) = (split4(k) + split4(k + 1) + split4(k + 2) + split4(k + 3) + split4(k + 4) + split4(k + 5) + split4(k + 6) + split4(k + 7)) Mod 2
k = k + 8
Next
If seed (1) = "0" And seed (2) = "0" And seed (3) = "0" And
Seed (4) = "0" And seed (5) = "0" And seed (6) = "0" And seed (7) = "0" And seed (8) = "0" Then stop
```

End If
End

Second proposal Experiment and results

The output sequence for the system check by the test measure randomization and the results.

Initial Seed "00101011"	Initial Seed "10111111"
 pass frequency test 0.4 <= 3.8415 pass serial test 1.1786 <= 5.9915 pass poker test 5.4150 <= 14.0671 pass runs test 1.5309< = 9.4877 pass autocorrelation test -0.159 <= 1.96 First round	 pass frequency test 1.225 <= 3.8415 pass serial test 3.6743 <= 5.9915 pass poker test 7.8301 <= 14.0671 pass runs test 4.0870< = 9.4877 pass autocorrelation test 0.3182 <= 1.96 First round
 pass frequency test 0.025 <= 3.8415 pass serial test 0.5976 <= 5.9915 pass poker test 4.5094 <= 14.0671 pass runs test 4.9014< = 9.4877 pass autocorrelation test 0.1591 <= 1.96 second round	third round  pass frequency test 0.625 <= 3.8415 pass serial test 4.2240 <= 5.9915 pass poker test 8.4339 <= 14.0671 pass runs test 7.5170< = 9.4877 pass autocorrelation test -0.159 <= 1.96
 pass frequency test 0.1 <= 3.8415 pass serial test 0.1704 <= 5.9915 pass poker test 8.7358 <= 14.0671 pass runs test 2.3849< = 9.4877 pass autocorrelation test 0.3182 <= 1.96 third round	 pass frequency test 0.1 <= 3.8415 pass serial test 0.1201 <= 5.9915 pass poker test 4.2075 <= 14.0671 pass runs test 2.1985< = 9.4877 pass autocorrelation test -0.795 <= 1.96 third round
 pass frequency test 0 <= 3.8415 pass serial test 1.8867 <= 5.9915 pass poker test 3.3018 <= 14.0671 pass runs test 2.3436< = 9.4877 pass autocorrelation test 1.2728 <= 1.96 fourth round	 pass frequency test 1.225 <= 3.8415 pass serial test 0.9070 <= 5.9915 pass poker test 11.754 <= 14.0671 pass runs test 4.7239< = 9.4877 pass autocorrelation test -0.477 <= 1.96 fourth round
 pass frequency test 0.9 <= 3.8415 pass serial test 2.6408 <= 5.9915 pass poker test 9.0377 <= 14.0671 pass runs test 4.4357< = 9.4877 pass autocorrelation test -0.636 <= 1.96 fifth round	 pass frequency test 0.625 <= 3.8415 pass serial test 0.6517 <= 5.9915 pass poker test 4.8113 <= 14.0671 pass runs test 3.3081< = 9.4877 pass autocorrelation test 0.9546 <= 1.96 fifth round

CONCLUSIONS

Through scientific study and application keys generators, noticing that not all keys generators have random features but based on the repetition of large parts of the original key that is leading to double standards statistical analysis of the key and retrieved. The traditional systems for generating keys don't have high level of complexity which leads to analyse easily through the prediction Keys by neural networks and genetic algorithms and other methods of analysis. While proposed systems have the ability to generate random keys with different lengths that we need in encryption. In addition to, the use of logic circuits can increase complexity.

REFERENCES

[1].David bishop, "introduction to cryptography with ™ applets", Grinnell College," Jones and Bartlett Publishers International", 2003

[2]. Oded Gold Reich, "Foundations of Cryptography", Springer-Verlag, 1999

[3].Michael Welschenbach, "Cryptography in C and C++", second edition, by Springer-Verlag, 2005

[4]. Information security. Retrieved from Wikipedia the free Encyclopedia:
http://en.wikipedia.org/wiki/Information _seaurity,2005.

[5].William Stallings, "CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE"
, FIFTH EDITION, 2011

[6].Shah Kruti R., Bhavika Gambhava,"New Approach of Data Encryption Standard Algorithm" , International Journal of Soft Computing and Engineering, 2012

[7]. Schneier, B.," Applied Cryptography: Protocols, Algorithms, and Source Code in C.",Second Edition. New York: John Wiley & Sons, 1996.

[8]. Andrew Rukhin, Juan Soto, James Nechvatal,Miles Smid, Elaine Barker, Stefan Leigh,"statistical test suite for random and pseudorandom number genertors for cryptographic applications",NIST Special Publication 800-22, May 2001.