Proposal Dynamic Keys Generator for DES algorithms

مولد مفاتيح ديناميكي لخوارزمية تشفير البيانات القياسية

DR. Alaa kadhim **B.Sc. Mohammed salih**

University of technology, baghdad

Proposal Dynamic Keys Generator for DES algorithms

مولد مفاتيح ديناميكي لخوارزمية تشفير البيانات القياسية

DR. Alaa kadhim **B.Sc. Mohammed salih**
University of technology, baghdad

Abstract:-

The symmetric algorithms encryptionisbasedon the key a symmetric between the sender and the recipient and the power of the sealgorithmsin preserving these keys from lack of access to people unauthorized through analysis for cipher text sencoded which transferred some proper ties explic it text, that of the fundamental problem sin the symmetric encryption algorithms is how to generate key srandomly feature ehighanaly stprevent people from retrieving symmetric secret keys.

In this paper, the basic idea is to design and implement dynamic keys generatorshave the ability to generate a large amount of randomness keys. In addition, the proposed models have the ability to generate keys of varying lengths and also be dynamic application

Keywords: cryptography, key generator, randomness, symmetric key, LFSRs.

1. Introduction

Historically, cryptography arose as a means to enable parties to maintain privacy of the information they send to each other, even in the presence of an adversary with access to the communication channel. While providing privacy there remains a central goal, the field has expanded to encompass many others, including not just other goals of communication security, such as

guaranteeing, integrity and authenticity of communications, but many more sophisticated and fascinating goals **[1].** Cryptographyis a discipline of mathematics and computer science concerned with information security and related issues, particularly encryption, authentication, and such applications as access Control **[2]**. Cryptography, as an interdisciplinary subject, draws on several fields. Prior to the early 20th century, cryptography was chiefly concerned with Linguistic patterns. Since then, the emphasis has shifted, and Cryptography now makes extensive use of mathematics, including topics from information theory, computational complexity, statistics, combinatory, and especially number theory **[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 **[2]**.

2. Feistel Mode

In cryptography, a **Feistel cipher** is a symmetric structure used in the construction of block ciphers, named after the German IBM cryptographer Horst Feistel. A large proportion of block ciphers use the scheme, including the Data Encryption Standard (DES). The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Therefore the size of the code or circuitry required to implement such a cipher which is nearly halved. Feistel networks and similar constructions are product ciphers, and so combine multiple rounds of repeated operations, such as:

- Bit-shuffling (often called permutation boxes or P-boxes).
- Simple non-linear functions (often called substitution boxes or S-boxes).
- Linear mixing (in the sense of modular algebra) using XOR to produce a function with large amounts of what Claude Shannon described as "confusion and diffusion". Bit shuffling creates the diffusion effect, while substitution is used for confusion **[4].**

3. Data Encryption Standard (DES)

Without doubt the first and the most significant modern symmetric encryption algorithm is that contained in the Data Encryption Standard (DES). The algorithm has been in wide international use, The Data Encryption Standard (DES), as specified in FIPS Publication 46- 3 is a block cipher operating on 64-bit data blocks**[4]**. 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. With another 56 bit derivative of the 64 bit key, the same process is repeated. Full details of DES are given in Algorithm (1) **[5, 6].**

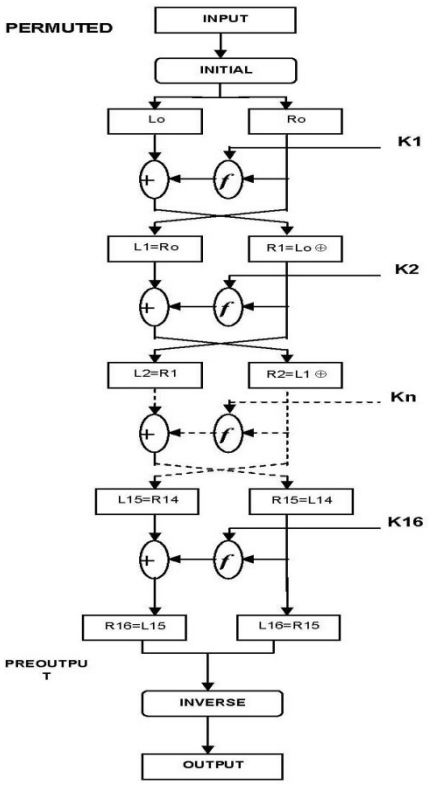


Figure (1): DES computation path

Algorithm (1): Data Encryption Standard (DES)

INPUT: plaintext m1 . . . m64; 64-bit key K=k1 . . . k64 (includes 8 parity bits).

OUTPUT: 64-bit cipher text block C=c1 . . .c64.

1. (Key schedule) Compute sixteen 48-bit round keys Ki, from K.

2. (L0, R0) ← IP(m1, m2,. . .m64) (Use IP Table to permute bits; split theresult into left and right 32-bit halves L0=m58m50 . . . m8,R0=m57m49 . . . m7)

3. (16 rounds) for i from 1 to 16, compute Li and Rias follows:

3.1. Li=Ri-1

3.2. Ri = Li-1 ⊕f (R i-1, Ki) where f(Ri-1, Ki) = P(S(E(Ri-1) ⊕ Ki)), computed as follows:

(a) Expand Ri-1 = r1r2 . . . r32 from 32 to 48 bits, T ←E(Ri-1).

(b) T' ← T ⊕ Ki . Represent T ' as eight 6-bit character strings: T '= (B1 . . . B8)

(c)T '' ← (S1(B1), S2(B2), . . . S8(B8)). Here Si(Bi) maps to the 4-bit entry in

row r and column c of Si

(d)T''' ← P(T''). (Use P per table to permute the 32 bits of T''=t1t2 . . . t32,

yielding t16t7 . . . t25.)

4. b1b2 . . . b64 ← (R16, L16). (Exchange final blocks L16, R16.)

5. C← IP-1 (b1b2 . . . b64).

4. 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

$$\mathbf{X_1 = (n_0 - n_1)^2 / n}$$

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

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

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] ≥ 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 i^{th} 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

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

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

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

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

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

$X_5=2(A(d)-(n-d))/(sqrt(n-d))$

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].**

5. 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.

5.1 First Proposal

In this's proposal designed system consist two part **(A, B).** The first part**(A)** have three linear feedback shift registers (LFSR), denoted by ***AR1, AR2, AR3*** with different length 8, 7, 6 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 $x^d + 1$ for any d that divides $2^n - 1$. In addition, the primitive feedback polynomials choose in part **(A)** are:

AR1=1+x^2+x^3+x^4+x^8

AR2=1+x^1+x^2+x^3+x^7

AR3=1+x^2+x^3+x^5+x^6

The second part**(B)** have three linear feedback shift registers (LFSR), denoted by **BR1, BR2, BR3** with different length 8, 7, 6 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^4+x^5+x^6+x^8

BR2=1+x^3+x^7

BR3=1+x^1+x^4+x^5+x^6

The outputs of part A and part B merge to provide sequence of bit passed to P-BOX (make permutation function) to represent random key for any block cipher, then selected bit (even index only) from result of The outputs of parts A, B to present new seeds for system A, B.

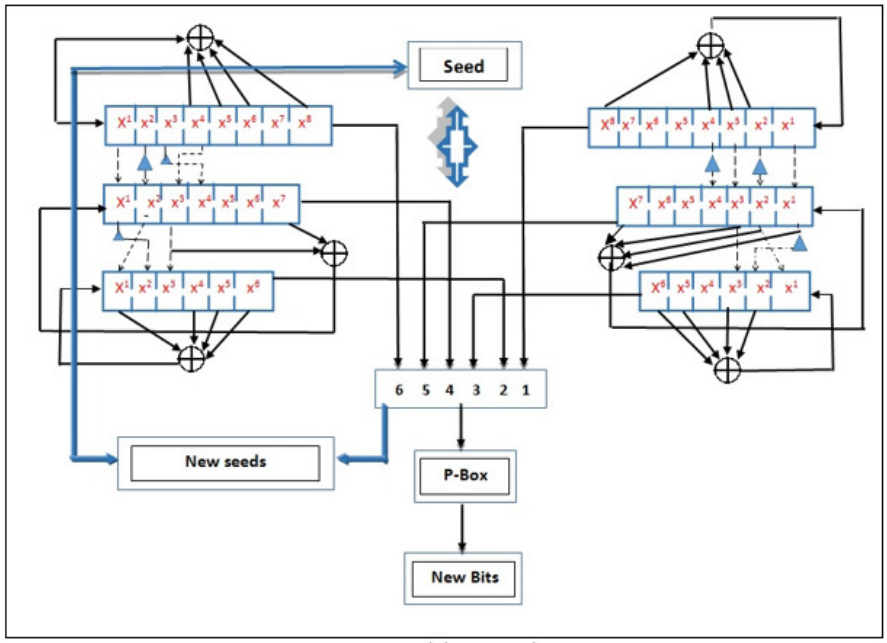


Figure (2): proposal 1

5.1.1First Proposal Algorithm

INPUT:

Different Initial Seeds to LFSRs of parts**A** and **B**.

OUTPUT:

Random sequence of**(0, 1)** used for DES block cipher

For part**A**

Max period 2^8-1 represent by **AR1** because **AR1** provide period largest than **AR2, AR3**

For i = 1 to max period do

Begin

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

For part**B**

Max period 2^8-1 represent by **BR1** because **BR1** provide period largest than **BR2, BR3.**

For j = 1 to max period do

Begin

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

- Make merge to arrays register systems in the fallowing form:

B1**concat**A2 **concat**B2**concat**A3**concat**B3**concat**A1

- Pass result of merge arrays to P-BOX (make permutation) to represent random key for any block cipher,
- Selected bit (even index only) from result of merge arrays to present new seeds for parts A, B.

5.1.12First proposal Experiment and results

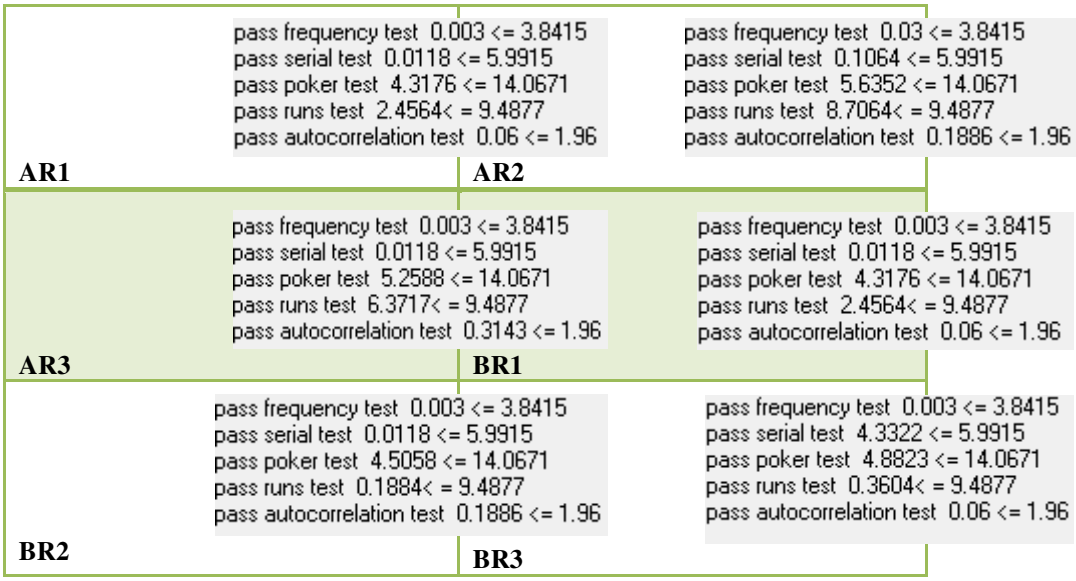
Initial Seeds to LFSR of part A

AR1: 10101111, AR2: 0000011, AR3: 001101

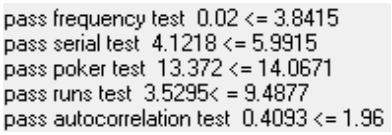
Initial Seeds to LFSR of system B

BR1: 11101101, BR2: 1110110,BR3: 011001

After programing and check the output for result parts A, B are randomize:



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



5.2 SecondProposal

In this's proposal designed cubic system consist six faces**(A, B, C, D, E, F)**as shown in the figure (3).The first face (A) have two linear feedback shift registers (LFSRs), denoted by ***AR1, AR2,*** with same length **(3)**but feedback of AR1 come from tapsequence of AR2 and viceversa. As shown in the figure (3.1).

In addition, the Primitive feedback polynomials choose in face**(A)** are:

AR1=1+x^1+x^3

AR2=1+x^2+x^3

The second face (B) have two linear feedback shift registers (LFSRs), denoted by ***BR1, BR2,*** with sameLength**(4)** but feedback of BR1 come

from tap sequence of BR2 and vice versa. As shown in the figure (3.2). In addition, thePrimitive feedback polynomials choose in face**(B)** are:

BR1=1+x^1+x^4

BR2=1+x^3+x^4

The third face (C) have three linear feedback shift registers (LFSR), denoted by **CR1, CR2, CR3** with same length 5 with Parallel shifting INPUT - linear feedback shift registers (PSI-LFSR) as shown in following figure(3.3). In addition, the primitive feedbacks polynomials choose in face**(C)** are:

CR1=1+x^2 +x^5

CR2=1+x^3+x^5

CR3=1+x^1+x^2+x^3+x^5

The fourth face (D) have three linear feedback shift registers (LFSR), denoted by **DR1, DR2, DR3** with same length 6 with Parallel shifting INPUT - linear feedback shift registers (PSI-LFSR) as shown in following figure(**3**.4). In addition, the primitive feedbacks polynomials choose in face **(D)** are:

DR1=1+x^1 +x^6

DR2=1+x^5+x^6

DR3=1+x^1+x^4+x^5+x^6

The fifth face (E) have seven feedback shift registers (LFSR) denoted by **ER1, ER2, ER3, ER4, ER5, ER6,ER7** with same

length 7 as shown in following figure(**3**.5). In addition, the primitive feedbacks polynomials choose in face **(E)** are:

ER1=1+x^1+x^2+x^4+x^5+x^6+x^7ER2=1+x^1+x^2 +x^7

ER3=1+x^5+x^6+x^7ER4=1+x^1+x^3 x^6+x^7

ER5=1+x^1+x^4+x^6+x^7ER6=1+ x^2+ x^5+x^6+x^7

ER7=1+x^1+x^3+x^6+x^7

The Sixth face (F) have eight feedback shift registers (LFSR) denoted by **FR1, FR2, FR3, FR4, FR5, FR6,FR7,FR8** with same length 8 as shown in figure(**3**.6). In addition, the primitive feedbacks polynomials choose in face **(F)** are:

FR1=1+x^2+x^3+x^4+x^8FR2=1+x^2+x^5+x^6+x^8

FR3=1+x^2+x^3+x^7+x^8FR4=1+x^1+x^3+ x^5+x^8

FR5=1+x^1+x^2+x^7+x^8FR6=1+ x^1+x^6+x^7+x^8

FR7=1+x^3+x^5+x^7+x^8FR8=1+x^3+x^5+x^6+x^8

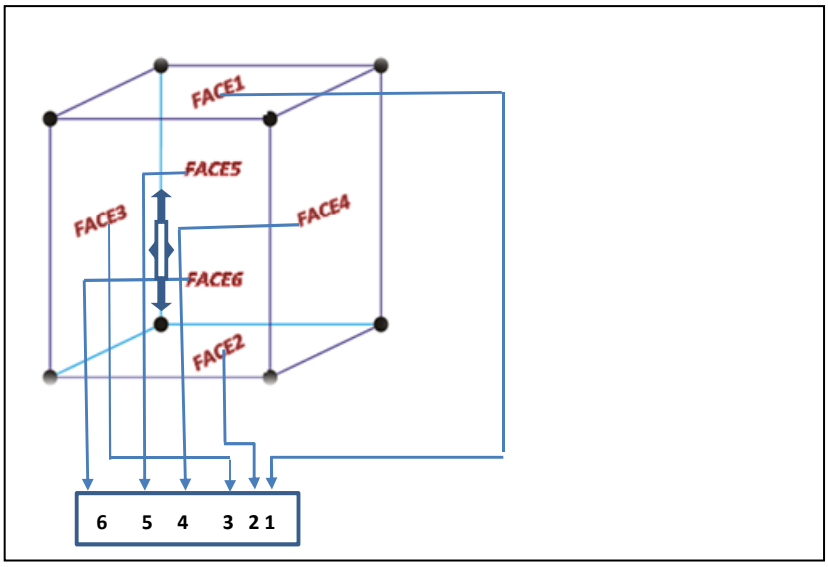


Figure (3): proposal Two

In the following figures are LFSRs for all face cubic (A, B, C, D, E, F)

FACE2
FACE6

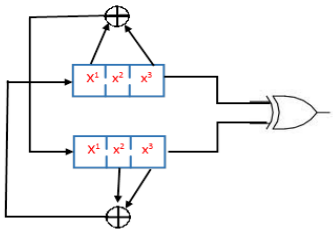


Figure (3.1): Face A

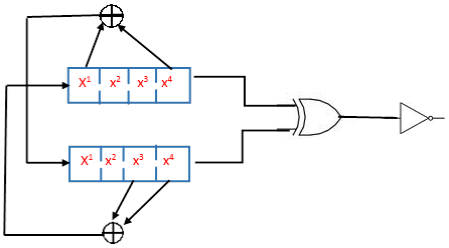


Figure (3.2): Face B

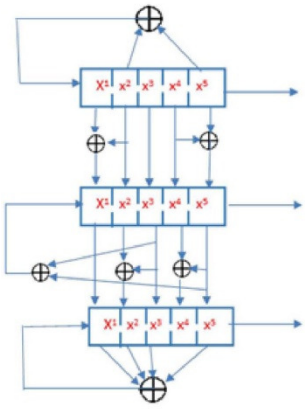


Figure (3.3): Face C

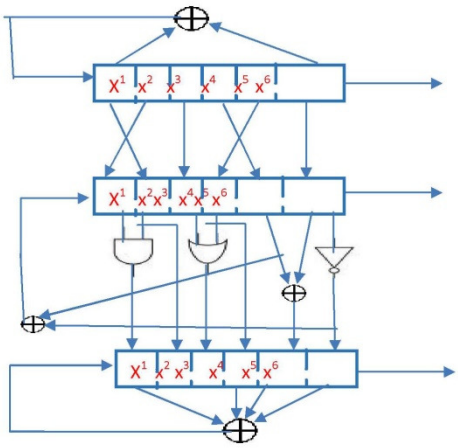


Figure (3.4): FaceD

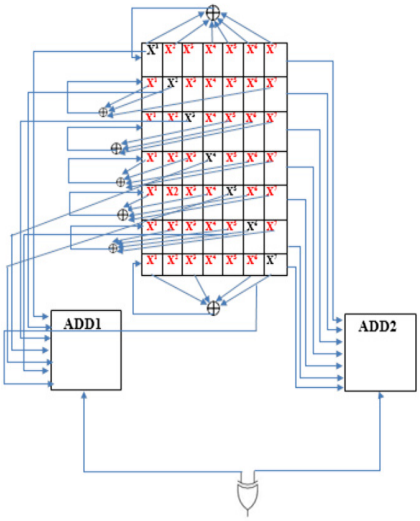


Figure (3.5): Face E

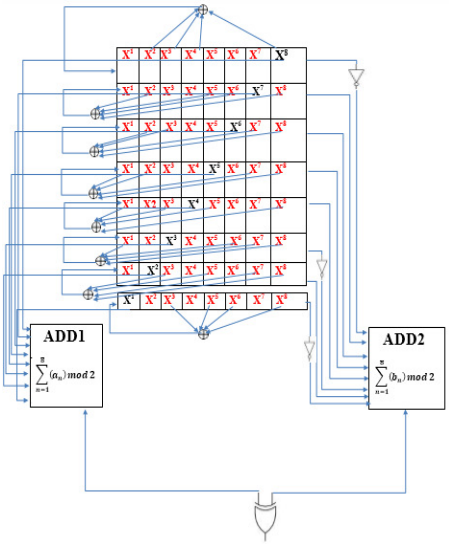


Figure (3.6): Face F

5.2 .1 Second ProposalAlgorithm

INPUT:

DifferentInitial Seeds to LFSRs of cubic system (six faces)

OUTPUT:

Random sequence of 0, 1 used for any block cipher system.

Process:

For face**A**

Max period 2^3-1 represent by **AR1**or **AR2** because **AR1** provide period equal to**AR2**

For i = 1 to max period do

Begin

Take outputs **AR1, AR2** respectively and store in two array **A1, A2** respectively with characteristic feedback of

AR1 come from tap sequence of AR2 and feedback of AR2 come from tap sequence of AR1.

Make xor operation between A1, A2

Store the output of face **A**

End

For face**B**

Max period 2^4-1 represent by **BR1**or **BR2** because **BR1** provide period equal to**BR2**

For i = 1 to max period do

Begin

Take outputs **BR1, BR2** respectively and store in two array **B1, B2** respectively with characteristic feedback of

BR1 come from tap sequence of BR2 and feedback of BR2 come from tap sequence of BR1.

Make xor operation between B1, B2.

Make NOT operation OF the result

Store the output of face **B**

End

For face**C**

Max period 2^5-1 represent by any resister in face**C** because providessame periodequal.

For j = 1 to max period do

Begin

Take outputs **CR1, CR2 and CR3** respectively and store in three array **C1, C2, and C3** respectively

Make Parallel shifting input

CR2(1) = CR1(1) Xor CR1(2),CR2(2) = CR1(2),CR2(3) = CR1(3),CR2(4) = CR1(4),CR2(5) = CR1(4) Xor CR1(5)

,CR3(1) = CR2(1),CR3(2) = CR2(2) Xor CR2(3),CR3(3) = CR2(3),CR3(4) = CR2(4) Xor CR2(5),CR3(5) = CR2(5)

Make shift to CR1, CR2, and CR3

Store the output of face **C**

End

For face**D**

Max period 2^6-1 represent by any resister in face**D** because provides same period equal.

For j = 1 to max period do

Begin

Take outputs **DR1, DR2 and DR3** respectively and store in three array **D1, D2, and D3** respectively

Make Parallel shifting input

DR2(1) = DR1(2),DR2(2) = DR1(1),DR2(3) = DR1(3),DR2(4) = DR1(5),DR2(5) = DR1(4),DR2(6) = DR1(6)

,DR3(1) = DR1(1) And DR1(2),DR3(2) = DR2(2),DR3(3) = DR2(3) Or DR2(4),DR3(4) = DR2(4)

,DR3(5) = DR2(4) Xor DR2(5),DR3(6) = flip(DR2(6))

Make shift to DR1, DR2, and DR3

Store the output of face **D**

End

For face**E**

Max period 2^7-1 represent by any resister in face**E** because provides same period equal.

For j = 1 to max period do

Begin

Put registers of face **E** in matrix

Take first diagonal of matrix and pass to **ADD1** function

Take outputs **ER1, ER2 ER3, ER4, ER5ER6and ER7** respectively and pass to **ADD2** function

Make xor operation between the result of **ADD1** function and the result of **ADD2**function

Store the output of face **E**

End

For face**F**

Max period 2^8-1 represent by any resister in face**F** because provides same period equal.

For j = 1 to max period do

Begin

Put registers of face **F** in matrix

Take Second diagonal of matrix and pass to **ADD1** function

Take outputs NOT (FR1)**, FR2 FR3, FR4, FR5**,NOT (FR6)**, FR7and** NOT (FR8) respectively and pass to **ADD2** function

Make xor operation between the result of **ADD1**function and the result of **ADD2**function

Store the output of face **F**

End

5.2.2Second proposal Experiment and results

Initial Seeds to LFSR of FACE A

AR1: 101 AR2: 011

Initial Seeds to LFSR of system B

BR1: 1001 BR2:0011

Initial Seeds to LFSR of FACE C

CR1: 01001 CR2: 00101 CR3: 10001

Initial Seeds to LFSR of system D

DR1: 000011 DR2: 110011 DR3: 100111

Initial Seeds to LFSR of FACE E

ER1: 110111 ER2: 1100001 ER3: 0000111 ER4: 1010011 ER5: 1001011 ER6: 010011 ER7: 1010011

Initial Seeds to LFSR of system F

FR1: 01110001 FR2: 01001101 FR3: 01100011 FR4: 10101001

FR5: 11000011FR6: 10000111 FR7: 00101011 FR8: 00101101

After programing and check the output for result cubic system (six faces) are randomize:

pass frequency test 0.1428 <= 3.8415 pass serial test 0.5238 <= 5.9915 pass poker test 0.1428 <= 14.0671 pass runs test 0.6944< = 9.4877 pass autocorrelation test 0.4472 <= 1.96 **Face A**	pass frequency test 0.06 <= 3.8415 pass serial test 1.3619 <= 5.9915 pass poker test 6.6666 <= 14.0671 pass runs test 0.7205< = 9.4877 pass autocorrelation test 0.8320 <= 1.96 **Face B**
pass frequency test 0.2688 <= 3.8415 pass serial test 7.9008 <= 5.9915 pass poker test 0.4347 <= 14.0671 pass runs test 0.1342< = 9.4877 pass autocorrelation test 0.7337 <= 1.96 **Face C**	pass frequency test 0.08 <= 3.8415 pass serial test 5.9278 <= 5.9915 pass poker test 3.8064 <= 14.0671 pass runs test 0.4519< = 9.4877 pass autocorrelation test 0 <= 1.96 Face D
pass frequency test 2.2755 <= 3.8415 pass serial test 2.8990 <= 5.9915 pass poker test 7.9047 <= 14.0671 pass runs test 2.0968< = 9.4877 pass autocorrelation test 0.8049 <= 1.96 **Face E**	pass frequency test 0.03 <= 3.8415 pass serial test 0.6103 <= 5.9915 pass poker test 10.905 <= 14.0671 pass runs test 3.3060< = 9.4877 pass autocorrelation test 1.0687 <= 1.96 Face F

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

pass frequency test 0.03 <= 3.8415
pass serial test 0.6450 <= 5.9915
pass poker test 12.939 <= 14.0671
pass runs test 4.0620< = 9.4877
pass autocorrelation test 1.2835 <= 1.96

6. Complexity Comparison between Proposal Dynamic Keys Generatosr and DES key Generator

The Complexity of DES key generator is($\mathbf{2^{56}}$)while the Complexity of proposal(**1**)is ($\mathbf{2^{672}}$)because it consist of two part denoted(**A, B**)is 2^{8*7*6} ,2^{8*7*6}respectively.The Complexity of proposal **(2)** is($\mathbf{2^{20160}}$) because The Complex ityof sixfacesdenoted **(A, B, C, D, E, F)**

7. Conclusion

Through scientific study and application generators keys we have noticed:

1- That not all features random generators, but on the repetition of large parts of the original key that isleading to double standards statistical analysis of the key and retrieved.

2- The traditional systems for generating keys don't have high level of complexity which leads to easily analyzed through the prediction Keys by neural networks and genetic algorithms and other methods of analysis. While systems proposed have the ability to generate random keys with different lengths that we need in encryption. In addition to the use of logic circuits change In the pattern of output data.

3- Used logical circuit in key generator system can get more complicity and efficient pseudorandom.

8. Suggestion

After the theoretical and predication we suggestion to used:

1- Used the artificial intelligent algorithm to design the good random generated.

2- Design the cloud key generator to generated keys and management all client.

الخلاصة:ـ

مولد مفاتيح ديناميكي لخوارزمية تشفير البيانات القياسية

إن خوارزميات التشفير المتناظر هي التي تعتمد على المفتاح المتناظر بين المرسل والمستلم وقوه هذه الخوارزميات في الحفاظ على هذه المفاتيح من عدم

الوصول إلى الأشخاص الغير مخولين وذلك من خلال عمليات التحليل التي يجيرها على النصوص المشفرة التي تنقل بعض خصائص النص الصريح, ان من المشاكل الأساسية في خوارزميات التشفير المتناظر هي كيفيه توليد مفاتيح تمتاز بعشوائية عاليه تمنع الأشخاص المحللين من استرجاع المفاتيح السرية المتناظر.

في هذا البحث الفكرة الأساسية هي تصميم وتنفيذ مولدات مفاتيح ديناميكية لها القدرة على توليد كم هائل من المفاتيح تمتاز بالعشوائية بالإضافة إلى أن النماذج المقترحة لها القدرة ان توليد مجموعات من المفاتيح متباينة الأطوال وتكون ديناميكية التطبيق

References;-

[1] Ali M.,"**Design public key cryptography base on new discrete logarithm problem**", PhD, thesis, Baghdad, university of technology, **2007**.

[2]William S.," **Cryptography and Network Security Principle and Cractice**", fifth edition,Prentice Hall**,**2011.

[3] SchneiderB.,"**Applied cryptography: protocols, Algorithms, and source code in C**.", second edition .New York: John Wiley & sons, 1996.

[4] "Feistel Cipher", Wikipedia, the free encyclopaedia Retrieved from:

http://www. Wikipedia.org

[5] Shah R., Bhavika G.,"**New Approach of Data Encryption Standard Algorithm**", International Journal of Soft Computing and Engineering,2012

[6]John T., "**Complexity and Cryptography an Introduction",**Cambridge University Press, 2006

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

[8] Andrew R., Juan S.,"**statistical test suite for random and pseudorandom number genertors for cryptographic applications**",NIST Special Publication 800-22, May 2001.