Tikrit Journal of Pure Science 22 (11) 2017 *ISSN: 1813 – 1662 (Print)*
E-ISSN: 2415 – 1726 (On Line)

Using permutation function and the Bank of LFSR's in pseudo random generator keys

Ali Khalil Salih
Mathematics Department , College Education for Women , University of Tikrit , Tikrit , Iraq
Email: ali.khalil.salih@gmail.com

Abstract
In this paper a Bank of Linear Feedback Shift Registers (LFSR) is used to design and build a system for cryptography. This algorithm consists of 16 Linear Feedback Shift Registers. When we choose eight Shift Registers for each operation of this system, we get one byte constantly this operation.
The external unit consists of three Linear Feedback Shift Registers use for filling the main system to generate the stream cipher of bits (0&1) as a Pseudo Random generator, in which we use the permutation system and this sequence of bits passed the statistical tests.

Key words: (LFSR) Linear Feedback Shift Register, Bit Permutation, Pseudo Random Generator, External Unit.

1. Introduction
One century ago when Gilbert Virnam discovered the stream cipher in 1917, and took the named afterwards. [1]
The types of algorithm use in cryptography representing in three types: the first one is Symmetric cipher, and the second type is Asymmetric cipher, finally the Protocols. The first Symmetric cipher is also divided into two kinds: block cipher in addition to stream cipher. [1]
The key use in encryption and decryption of Symmetric algorithm is same in both operations.
Encryption is defined as the process to convert the plain text (input text) into the cipher text (output text). [2]

Decryption is the process to convert the cipher text into the plain text. Symmetric cryptography separate between the block cipher and the stream cipher, it is easy to differentiate and describe the operational differed between stream cipher and block cipher in encryption n bit at a time, where n represents the width in Stream cipher, it encrypts n bits individually, this is attained by adding the bit from key to plain text bit . The stream cipher is synchronous if the key stream depends on key, but asynchronous if the key stream depends on the cipher text only. [1]

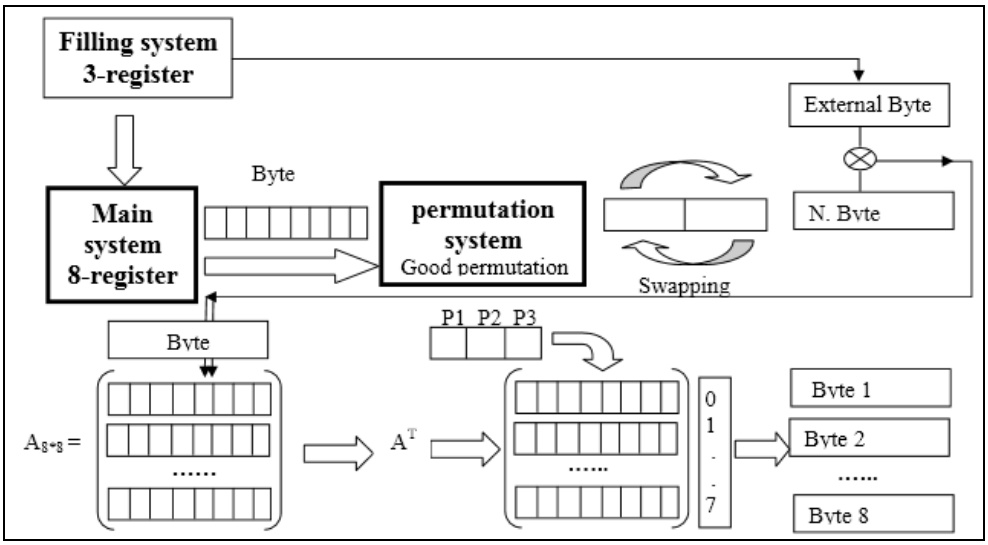

Fig (1:1) Flowchart for algorithm

2. Architecture of the proposed system
In the figure 1, the system consists of the following parts:
2.1. External unit: it uses for the purpose of filling special places of main system, which consists of three linear shift registers. The length for each shift register is composed of primary number of stages to ensure the completion of the long period.
2.2. The main system: consists of 16 linear shift registers, eight of them are selected for each operation of the system, and the outputs represent one byte, sends the one by one sequential to the next operation.

2.3. Permutation system: it is a system composed of matrix size 8*8. The bytes replace among positions for all bits, after that the outputs represent one byte for each operation, this byte will swap between two parts, and each part which consists of 4bits, that will change between first and second parts.

2.4. Matrix system: the new matrix size 8*8 that is generated from operation of permutation, and makes transpose for this matrix. In this matrix chooses one row from it represented one byte (8-bits) it is selected by three bits to determine the specific position.

3. Bit permutation

Bit permutation is attractive and a major important cryptography and also useful for many application that use in several algorithm such as (DES, Serpant, Two-fish) [3]

The Permutation is defined as ways to arrange or select small or equal number or object from any groups or collections of objects with suitable concern being paid for ordering of arrangement or selection.

The factor represents the Permutation of number of n things chosen r at the time given by the product, Permutation is an order list of finite set of elements. [4]

There are several formulas to write the permutation in mathematics depending on repetition

$$p_n^r = n(n-1)(n-2)...(n-r+1)$$
$$p(n,r) = n*(n-1)*...(n-(r-1))$$
$$p(n,r) = n!/(n-r)!$$
$$p(n,n) = n!$$
$$n! = 1*2*3*...*n$$
$$n! = n(n-1)(n-2)*...*2*1$$

3.1. Some types of bit permutation methods.[3]

The Permutation is used for building instruction such as:

GRP (Group Register Permutation) instruction is unaccustomed permutation R1&R2 source registers and destination is third register.

PPERM3R instruction obviously indicates the original position for each bit to destination register and uses **lg(n)** to specify position for one bit from **n** position.

PPERM modified version of **PPERM3R** except bits are not updated set to zero and combine result with OR instruction.

CROSS is based on Benes network formed by serial a butterfly network with inverse butterfly network number of bit consisting of **lg(n)** stages.

OMFLIP instruction alternative solution based on omega-flip network .

SWPERM with SIEVE instruction explicitly specific position of **bits** in the source register the SIEVE instruction works locally within each 4-bit subword .

4. Filling system

Filling the two systems is as follows

4.1. External unit: Filling is as in below

It Processes the Basic Key (BK) with the Message Key (MK) to fill the system which consists of three shift registers one by one, and fill the last position of each shift register by one.

4.2. Main system

After generate one byte by the external unit, the first four bits get address to determine the sequence of one shift register from sixteen and choose with seven LFSR after it. As we mention we need eight linear shift registers in each operation of the system. We use eight LFSR to run the main system and we obtain the last one by fill the last position of each shift registers by one.

The eight bits for each byte from external unit which are filled vertically sequentially. The final vertical position bit is got the value one to avoid zeros value for the one shift register.

5. Motion of the system

The system moves according to the following

5.1. The external unit moves to get four bits to select the shifts in the main system. After that it obtains the stream for bits to fill the main system, also gets from them for External Byte (EB). This means that the motion continues during the operation system.

5.2. After choosing the shift registers from the main system and filling it, it moves this system to obtain for byte enters to permutation system to get the permutation byte .which is divided into two parts each part consists of 4-bits swap and replaces the right part with the left part to get Swapping Byte.

5.3. Collecting Swapping Byte by gate XOR with External Byte (EB) in (1) above to get for New Byte (NB).

5.4. After getting 8 bytes from movement above, it is puts in matrix (8*8) then takes the Transpose Matrix(TM) for this matrix.

5.5. Getting the final byte from (TM) during three bits choosing the location in shift registers chosen from main system to get 64 bits first so on …until getting the stream of bits which is consider as a key pseudo random that gets it.

6. Testing and Results

The final process is to check the results obtained of output from this system, testing the stream bits generated from key generator through statistical test standard during take the sample there at least 512 bits, we shall choose the length of sequence is enough to make the statistical. [5] [6] [7] [8]

11001011 10010101 00110001 10100100 10101111
00010111 10000110 11001110 01100011 01001011
10001101 10101110 00010011 11000001 11000110
11000110 01101111 00001011 10110111 00100011
10000011 01110001 10100010 01010110 01110010
10001111 01000110 11001011 10000100 10110011
00100100 01101101 00101100 11101101 10110001
11100001 01001010 11011001 11001000 11011010
01111000 01111001 11111101 10000011 10000111
00011000 01011101 01101000 11001100 10001111
01101101 00111010 11101100 01010011 01010000
11101100 10111101 10011010 01010110 01001001
01000110 11101010 01101001 11000101

6.1. Frequency (Monobits) Test

p.Value for Frequency test = 0.7237

Sequence passes NIST frequency test for randomness.

6.2. Frequency Test within a Block (block length = 8)

p.Value for Block test = 0. 0.9943

Sequence passes NIST block test for randomness.

6.3. Runs Test

p.Value for Runs test = 0. 0.1830
Squence passes NIST runs test for randomness.

6.4. Maurers Universal Test

x = 3 The length of each block chisq = 1.994872 p = 0.849854 x = 4 The length of each block chisq = 1.420137 p = 0.922098
J too small (J < 500) for result to be reliable
success = True
plist= [0.11440059571878829, 0.012518757137198433, 0.36276892358840962, 0.27676736287949222, 0.50528858712893099, 0.63010448334135161, 0.84985409438650006, 0.92209784399624284]
sum = 7.16992500144
fn= 1.19498750024 sum of the log2 distances between matching x-bit templates
success = True
p = 0.767189261865
sum = 274.139100041
fn = 1.6715798783
success = True
p = 0.907678498069

6.5. Linear Complexity Test

M = 7 The length in bits of a block.
N = 9 The length of the bit string.
K = 6 The number of degrees of freedom.
chisq = 5.66666666667
P = 0.461545923305
L = 4 The length of the shortest linear feedback shift register.
p = 0.461545923305
sh-4.3$ python main.py
M = 7
N = 48
K = 6
chisq = 2.33333333333
P = 0.886634179201
L = 4
p = 0.886634179201

6.6. Serial Test

delta2 = 0.8
P1 = 0.808792135411
P2 = 0.670320046036
success = True
plist = [0.80879213541099859, 0.67032004603563855]
psi_sq_m = 13.5714285714
psi_sq_mm1 = 3.2619047619
psi_sq_mm2 = 0.107142857143
delta1 = 10.3095238095
delta2 = 7.15476190476
P1 = 0.0355243881192
P2 = 0.0279488017253
success = True
plist = [0.035524388119194839, 0.027948801725339617]

6.7. Approximate Entropy Test

Pattern 4 of 4, count = 79
phi(2) = -3.683985
Pattern 1 of 8, count = 27
Pattern 2 of 8, count = 46
Pattern 3 of 8, count = 42
Pattern 4 of 8, count = 50
Pattern 5 of 8, count = 46
Pattern 6 of 8, count = 46
Pattern 7 of 8, count = 50
Pattern 8 of 8, count = 29
phi(2) = -4.360244
AppEn(2) = 0.676259
ChiSquare = 11.3490932407
success = True
p = 0.0229085439353

6.8. Cumulative Sums Test

Pattern 7 of 8, count = 50
Pattern 8 of 8, count = 29
phi(2) = -4.360244
AppEn(2) = 0.676259
ChiSquare = 11.3490932407
success = True
p = 0.0229085439353
PASS
success = True
plist = [0.21919399348562651, 0.11486621530252161]
PASS
success = True
plist = [0.99237259942970879, 0.89043492560082971]

6.9. Random Excursion Test

success = True
plist = [0.99449693971811115, 0.98799664847827962, 0.96254157254546502, 0.96258996669000596, 0.50252874244726498, 0.14309859788329643, 0.98799664847827962, 0.99449693971811115]
J=26
x = -4 chisq = 8.869100 p = 0.114401
x = -3 chisq = 14.540194 p = 0.012519
x = -2 chisq = 5.456233 p = 0.362769
x = -1 chisq = 6.314954 p = 0.276767
x = 1 chisq = 4.312982 p = 0.505289
x = 2 chisq = 3.455678 p = 0.630104
x = 3 chisq = 1.994872 p = 0.849854
x = 4 chisq = 1.420137 p = 0.922098
J too small (J < 500) for result to be reliable
success = True
plist = [0.11440059571878829, 0.012518757137198433, 0.36276892358840962, 0.27676736287949222, 0.50528858712893099, 0.63010448334135161, 0.84985409438650006, 0.92209784399624284]

7. Conclusion

The results by use the two systems one is external and another is main system contents it change in every operation for system, that means increase in complexity and security for system, also the results are not constant means reality and certified, addition to unpredictable it in any time. We use a new system

in every run or operation, and the result passed the standard statistical tests, therefore this system is competent and suitable to use.

References

[1] C. Paar, J. Pelzl, **"Understanding Cryptography –A Textbook for Students and Practitioners",** Springer-Verlag Berlin Heidelberg, 2010.

[2] C. Paar, J. Pelzl, **"Understanding Cryptography Stream Cipher",** Springer-Verlag Berlin Heidelberg, 2010.

[3] ZJ Shi "**Bit Permutation Instructions: Architecture, Implementation, and Cryptographic Properties**"- Princeton University, Princeton, NJ, 2004.

[4] K.H. Rosen, **"Discrete Mathematics and Its Applications"**, 5th Edition, McGraw-Hill, 2003.

[5] A. Menezes, P. van Oorschot ,S. Vanstone, **"Handbook of Applied Cryptography",** CRC Press, Inc, 1996**.**

[6] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo''**a Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications'',** NIST Special Publication, 2001.

[7] https://repl.it/languages/csharp. https://msdn.microsoft.com/en-us/magazine/msdnmaggov13.

[8] https://www.tutorialspoint.com/execute_ python_ online.php.

https://github.com/dj-on-ithub/sp800_22_tests.

استخدام دالة التباديل ومصرف الزواحف الخطية لتوليد مفاتيح شبه عشوائية

علي خليل صالح

قسم الرياضيات ، كلية التربية للبنات ، جامعة تكريت ، تكريت ، العراق

Email: ali.khalil,salih@gmail.com

الملخص

في هذا البحث استخدمنا مصرف من الزواحف الخطية لتصميم وبناء خوارزمية التشفير, قمنا باختيار 8 من الزواحف الخطية من بنك الزواحف المكون من 16 من الزواحف الخطية كنظام رئيسي والذي يتم توليد سيل من الثنائيات (البت) واحد بعد الاخر بالتعاقب الوحدة الخارجية المكونة من 3 زواحف خطية تقوم بملئ المنظومة الرئيسية لتوليد سيل مشفر من البيانات البت (0,1) كمنظومة شبه عشوائية لتوليد المفاتيح التي يستخدم فيها منظومة التباديل , وهذه السلسلة من البت اجتازت معظم الاختبارات الاحصائية.