Implementing A New Serial Control As Nonlinear Function for Key Stream Pseudo-Random Number Generator

Ahmed Mohamed Mal -Allah Al-Suffar
ahmed_malallah@yahoo.com
Al-Iraqia University - College of Engineering
Software Engineering Department

Abstract: *Nonlinear behavior seems to be the rule rather than exception in nature, so the progressing in security and cryptography scope is very wide needed in the world.*
Problem: Cryptographically, there is potential needs to produce the nonlinear products or to complicate the linear system constrains to be as nonlinear system.
Problem Solving:Hence, to solve this problem a new nonlinear scheme for general-purpose pseudorandom number generator (Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator) is presented by this research which comprises two parts; the traditional part, as a combination of LFSR, and a serial part as suggested part comprising a new

approach of high complexity performance to meet standard cryptographic criteria.

Results: practically, a strong cryptographic key stream pseudo-random number generator was obtained in this research which provides a substantial increase of complex nonlinearities, leads to the best development of randomness statistical property output sequences, passed all statistical tests successfully, and has large linear equivalence to prevent correlation attacks.

Keyword: nonlinear product, cryptographic key, controllers, new nonlinear part, Key Stream Pseudo-Random Number Generator (PRNG), nonlinear system, Bits control Bar, Serial part.

1. Introduction

The generation of random numbers is essential to cryptography. One of the most difficult aspect of cryptographic algorithms is in depending on or generating, true random information. This is problematic, since there is no known way to produce true random data, and most especially no way to do so on a finite state machine such as a computer.

So, there are two kinds of random number generators: non-deterministic random number generators, sometimes called "true random number generators" (TRNG), and deterministic random number generators, also called pseudorandom number generators (PRNG), [6] as in this research.

Data is stored and transmitted by electronic means, there-by it will be vulnerable. The privacy of individuals as well as of organization depends severally on possibility of protecting communicated information from unauthorized disclosure and modification by an interceptor [1]. Cryptographic system provides secrecy by using the key to transform plaintext in to cipher text, which is concealed to any opponent. The reality of stream ciphers security lies somewhere between the simple XOR and One-Time pad. Any key stream pseudorandom number generator for practical stream cipher applications can generally be represented as an autonomous finite-state machine of secret key. Therefore, the security of a cryptosystem should rest in the key, not in the details of the algorithm; otherwise, if anyone uses a weak process to generate keys, then cryptographically the whole system is weak[4].

These are the motivations to propose this research; the Serial Control Key Stream Generator as a new technique, which is hardly analyzable, concerning the high degree of nonlinearity used. Finally, standard cryptographic criteria such as a large period, a high linear complexity, and good statistical properties are thus relatively easily satisfied. Such a generator may in principle be invulnerable to various methods of attacks.

1.1 Stream Ciphers

Divide the plaintext into characters to encipher each character with a time-varying function whose time-dependency is governed by the internal state of the stream cipher. After the enciphering each character, the device changes state according to some rule. The device which emits such a random sequence; i.e. a sequence where each bit is equally likely to be (0) or (1) independently of preceding bits, looks like a fair coin tossing sequences, is called a Binary Symmetric Source (BSS) .[2,7]

1.2 Key Stream Generators

The running key generator, which controlled by true key (base key) has the task to simulate a random sequence which is used to encipher the plaintext. Stream ciphers convert plaintext to cipher text by (1) bit at time. The simplest implementation of stream cipher is shown in Figure (1) below. A key stream generator outputs a stream of bits (kn). As in Eq (1) the key stream is XORed with a stream of plaintext bits (P_n) to produce the stream of cipher text bits (C_n). [4]

Then $C_n = P_n \oplus K_n$ ....Eq (1)

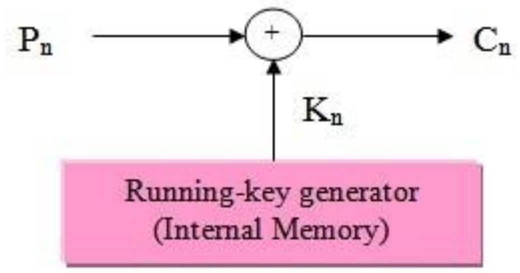


Figure (1) Stream Cipher

Cryptographically, running key generator is an autonomous finite state machine, it consists of an output alphabet and a state set, together with two functions and an initial state. The system's security depends entirely on the insides of the key stream generator which generates a bit stream that looks random, but is actually a deterministic stream that can be flawlessly reproduced at decryption time.

A key stream generator has three basic parts; Internal state describes the current state of the key stream generator. The output function takes the internal state to generate a key stream bit. While the next state takes the internal state and generates a new internal state. [10]

The basic problem of key stream generator design in the context of finite state functions and output function, which are guaranteed to produce a running key that satisfies large linear complexity and uniform distribution properties.

The construction of secure running key generator implies the introduction of nonlinear transformation by:

1. Implicit methods :By letting one LFSR clocks the second.
2. Explicit methods: By directly applying a nonlinear function, so the next state and the out mapping are candidates for nonlinear transformations. [9]

2. The Proposal

This proposal is one of deterministic random number generators, also called pseudorandom number generators (PRNG), considered every possible means of attack and protected against them all. This Generator has consideration of periodicity, Randomness and Non-linearity to be analyzed.

Thereby, Periodicity was determined explicitly to guarantee property of the sequence. Randomness in physical systems is usually attributed to external noise. But it is argued here that even without such random input, the intrinsic behavior of nonlinear system can be computationally so complicated as to seem random in all practical experiments. Then attention to statistical analyses was considered to prove randomness.

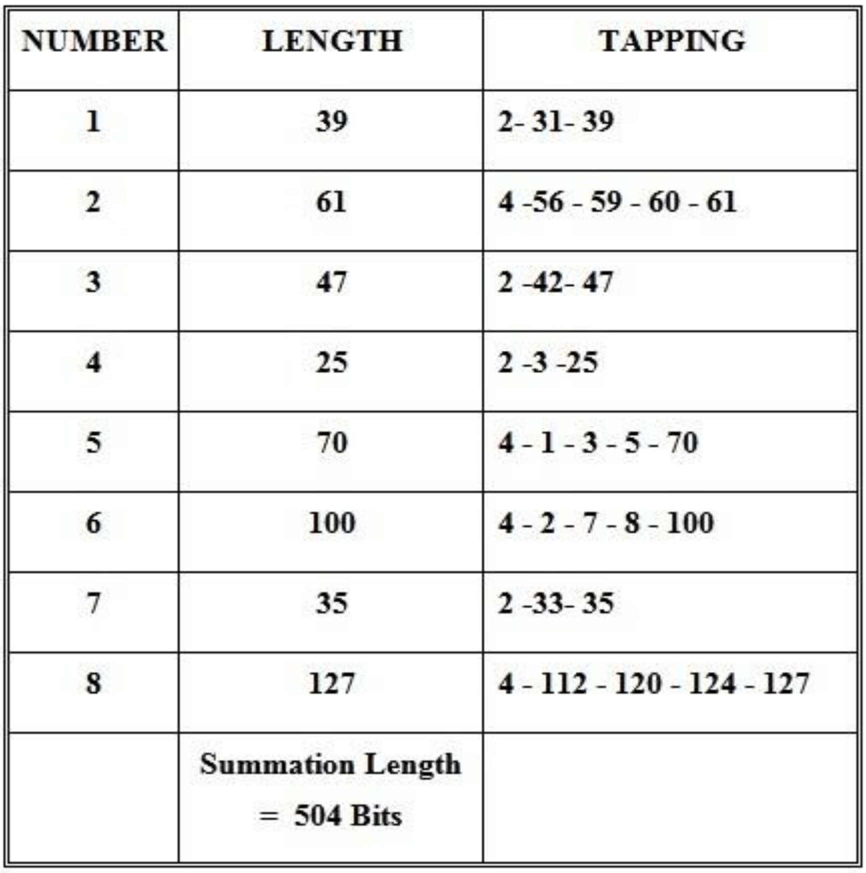
2.1 Experimental Work

In the first instance it is important to discuss the proposal as a practical process to generate keys by the suggested algorithm (Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator), as a running-key generator.

Here in, different feedback function polynomials were employed in proposed generator, provided that their lengths are all relatively prime, each of them has a maximal period and the feedback polynomials are all primitive that implies the whole generator output sequence to be as maximal length. As the output of the shift register is one bit at time (n), the period of each shift register is the length of the output sequence of the individual LFSR before it starts repeating.

The feedback function, as shown in Table (1), is simply the XOR of certain bits (tap sequence) in the LFSR. If it is initialized with random bits (0,1) it will produce a string of feedback function bit as output sequence that depends on exact tapping of internal states.

Table (1) Lengths and Tapping of LFSRs



NUMBER	LENGTH	TAPPING
1	39	2- 31- 39
2	61	4 -56 - 59 - 60 - 61
3	47	2 -42- 47
4	25	2 -3 -25
5	70	4 - 1 - 3 - 5 - 70
6	100	4 - 2 - 7 - 8 - 100
7	35	2 -33- 35
8	127	4 - 112 - 120 - 124 - 127
	Summation Length = 504 Bits	

For each particular LFSR of them, in order to be a maximal-period, the polynomial formed from a tap sequence plus the constant 1 must be a primitive polynomial mod 2. The degree of the polynomial is the length of each LFSR.

As in Fig (2) below the provisions have been made for two different configurations that are needed to explain the overall proposal algorithm; Part one is a common part of LFSRs, which comprises (8) LFSRs each (4) of them work separately to give their outputs as an address of nonlinear combined functions (EPROM1, EPROM2). Part two is a newly proposed part that uses (8) LSRs with no feedback. All work serially forward Input/output bit, controlled by (8) controllers to determine which bit will be selected.

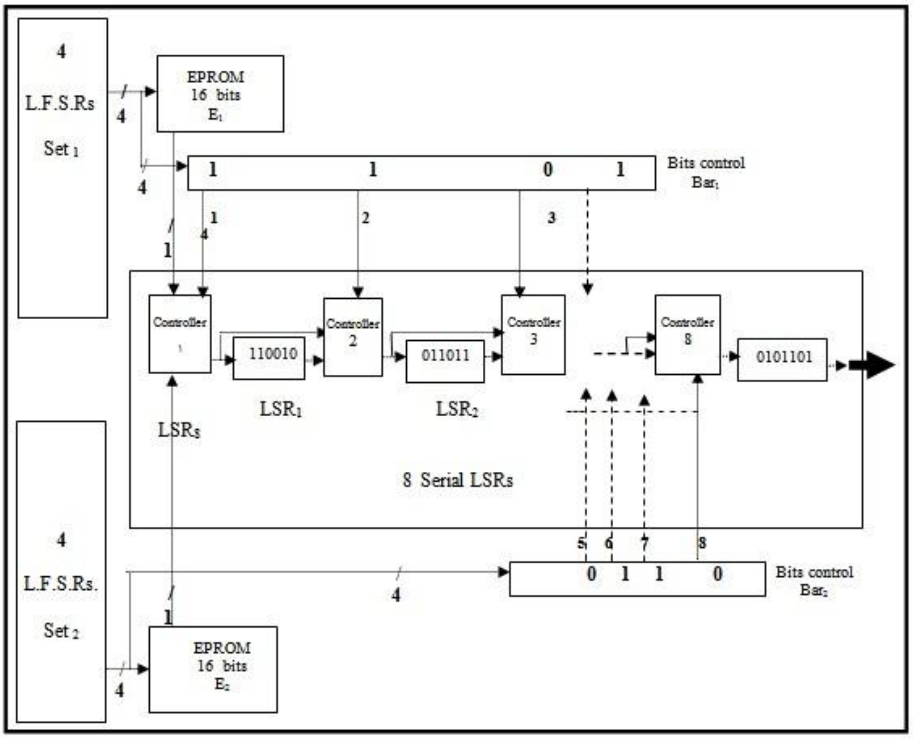


Fig. (2) the Block Diagram of Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator

2.1.1 Part one (Traditional Part)

This part consists of two divisions, as in Fig (3):

A. (8) LFSRs (Driving division), composed of two sets, each set comprises (4) LFSRs.

B. Two EPROMs (E1, E2)

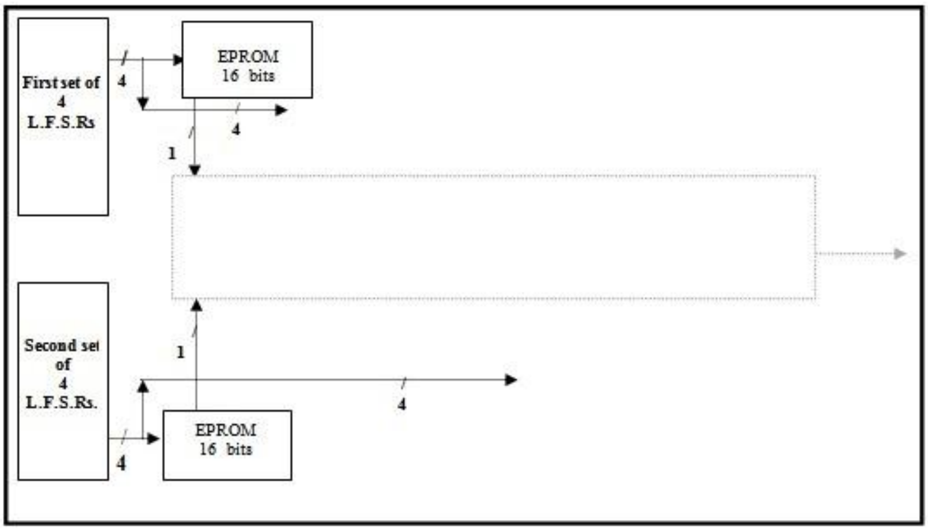


Fig (3) The Block Diagram of part one (Traditional Part) of Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator

2.1.1.1 (8) LFSRs (Driving Division)

In this research, basically LFSRs, as a branch of cryptography, were used to participate the design of this generator. It is proven that there is not susceptible to probabilistic correlation attacks; to give a lot of security with only a few logic gates.

However, in this research eight LFSRs with different lengths [as shown in Tab (1)] and different feedback function polynomials were employed, provided that their lengths are all relatively prime. Each LFSR of them, in order to have a maximal-period, the polynomial, to be a primitive polynomial mod 2, have to formed from exact tapping sequence (as in literature) plus the constant 1.

The degree of the polynomial is the length of the shift register. A primitive polynomial of degree n is an irreducible polynomial that divides ($X^{2^n-1}+1$) each of them has a maximal period and the feedback polynomials are all primitive that implies the whole generator output sequence to be as maximal length. As the output of the shift register is one bit at time (n), the period of each shift register is the length of the output sequence of the individual LFSR before it starts repeating.

In each LFSR the feedback function, as tapping sequence is simply the XOR of certain bits in the LFSR. If it is initialized with random bits (0, 1) it will produce the output sequence that depends on exact tapping of internal states. The output sequence of individual LFSR is the string of feedback function bit.

For instance, LFSR of (61-bits) length must be in one of (2^{61}-1) internal states to generate (2^{61}-1) bits-long pseudo-random

sequence before repeating. Note that only the LFSR of (61-bits) length with certain tap sequences will cycle through all (2^{61}-1) internal states before repeating; these are the maximal-period LFSRs and the resulting output sequence is called m-sequence.

Here in, to determine the primitive polynomials mod 2 , the choosing of a random polynomial, for example the Listing (61,5,2,1,0), must be tested by mathematical software Package to ensure whether it is primitive or not and the resultant LFSR has a maximal length; in other words the resultant sequence has 261-1 values before repeating itself.

As mentioned before, the proposal suggests (8) LFSR, with different lengths, to be implemented as two sets (first set, second set), each set consists of (4) LFSRs. So at each time (4) bits comes as output from each set of LFSRs to play its important role in two directions:

1. The first role is to compose the right address of the bit position in each EPROM.
2. The second role is to be as a controller to select the individual bit in Serial LSRs (Suggested part) in this research as it will be discussed.

In cryptography, it is necessary and important to equilibrate between the dense primitive polynomials and sparse primitive polynomials, where sparse feedback polynomial is not costly in hardware but it facilitates correlation attacks. However, in this research the different LFSRs with different primitive polynomials, each one with its exact values as listed in Table (1), were picked from the literature.

2.1.1.2 Nonlinear Combined Functions -EPROMs (E1, E2)

In order to avoid the potential danger inherent in a random selection of bits stored in EPROM, equilibrium of (0 and 1) bits were needed to prevent the likelihood of degeneracy to occur.

Cryptographically, designing EPROM as nonlinear function the following requirements have been considered to transfer the statistical properties of nonlinearity.

1. To maximize the running key complexity relatively to the nonlinear complexities.
2. It must be easy and fast to be implemented.
3. No leak of weakness must be exist, i.e. prevent any modularization attack (divide and conquer) directed towards the subsystems of the driving part of the key stream generator.

Using the EPROMs (E_1, E_2) needs (4) bits for each one of them as address to indicate the individual bit in it. So both (E_1, E_2) need (8) bits.

This proposal suggests (8) LFSRs as in Fig (3) to be implemented as two sets (first set, second set), each set consists of (4) LFSRs. At time (t) each LFSR has one output bit. So at time t (4) bits comes as output from either first set or second set of LFSRs to play its important role in two directions: the first one is to be the right address of the bit position in each EPROM and the second role is to be as a controller to select individual bit in Serial LSRs part in this research as it will be discussed.

2.1.2 Part two (Serial part; Suggested Part)

As in Fig (4), the proposal has a suggested part design consists of (8) controllers and (8) serial LSRs with different lengths, without tapping function or feedback.

Anyway, all the components in part two are arranged within two sets (set1, set2). Each set contain four controllers and four serial LSRs...

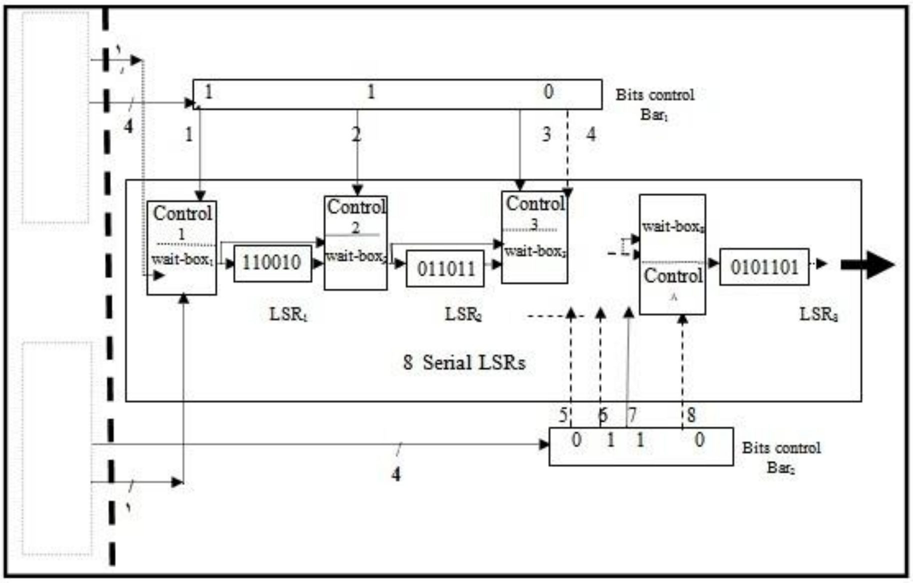


Fig (4) The Block Diagram of part two (Serial Part, Suggested Part) of Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator

Thereby, Set1 composed of (4) Controllers (Controller1, Controller2, Controller3, Controller4) dedicated to control the input bit to (4) serial linear shift registers (LSR1, LSR2, LSR3, LSR4) with different length of (100, 87, 94 and 98) respectively.

Set$_2$, also composed of (4) Controllers (Controller$_5$, Controller$_6$, Controller$_7$, Controller$_8$) dedicated to control the input bit to (4) serial linear shift registers (LSR_5, LSR_6, LSR_7, LSR_8) with different length of (84, 96, 88 and 91) respectively. Then the total length of all LSRs within two sets is 734 Bits .

So, the non-linearity configuration, which adds difficulties, was embodied by two means (EPROMs and Controller) as following:

1. The first means is the non-linearity imbedded in both (EPROMs) in set1 and set2 as a combine function.
2. The second means of the non-linearity is the function of the bit comes from (Bits control Bar) to determine the function of the individual controller. So by the (bit1) comes from (Bits control Bar) to (controller1) will select one of the bits comes from either (EPROM1) or (EPROM2) to be the input for the (LSR1). While, by the (bit2, 3, 4,...,8) comes from (Bits control Bar) to determine the function of (controller2), (controller3),..., and (controller8) to select the individual bit either from the input bit or output bit of previous (LSR) to be the input of the particular (LSR) of exact step. [,e 'g ... by the (bit2) comes from (Bits control Bar) determines the function of (controller2) to select the individual bit either from the input bit or output bit of (LSR1) to be the input of the (LSR2) and so on]

2.2 The Proposed Algorithm

1. To start implementing the system, each EPROM must full of equitable bits (0s, 1s).
2. Initiate the system by (250) bytes as an initial secret key; means it is (2000) bits to cover:
   a. (504) bits the needs of different lengths of (8) LFSRs (Driving division) as in Table (1).
   b. (738) bits is needed for (8) arbitrary serial LSRs (Suggested Part).
3. Let the system be Run.
4. Let the (4) bits of the output of the (First set of LFSRs) and the (4) bits of the output of the (Second set of LFSRs) be the

addresses of [EPROM (E_1) and EPROM (E_2)] respectively to yield one bit output from [EPROM (E_1)] and one bit output from EPROM (E_2)].

5. Let the (4) bits of the output of the (First set of LFSRs) be put in (Bits Controller Bar1) to use as a controller (Controller1, Controller2, Controller3, Controller4) to determine the individual Input bit from bits available in wait-boxes, as in Fig (4), to (4) serial linear shift registers (LSR_1, LSR_2, LSR_3, LSR_4) respectively.
6. Let the (4) bits of the output of the (Second set of LFSRs) be put in (Bits Controller Bar2) to use as a controller (Controller5, Controller6, Controller7, Controller8) to determined the individual Input bit from bits available in wait-boxes, as in Fig (4), to (4) serial linear shift registers (LSR_5, LSR_6, LSR_7, LSR_8) respectively.
7. Firstly, the two output bits yield from (step 4) being put in (wait-box1), the competition between these bits are settled by (control-bit) of controller1. If the bit of controller1 is (0) then the bit comes from (E_1) will be the input of LSR1, but If the bit of controller1 is (1) then the bit comes from (E_2) will be the input of LSR_1. Simultaneously, the same bit will inter to (wait-box2).
8. The output bit yield from LSR_1 also inter to (wait-box2).
9. The competition between two bits in (wait-box2) settled by bit controller2. If the bit of controller2 is (0) then the bit comes from (wait-box1) will be the input of LSR_2, but If the bit of controller2 is (1) then the bit comes from LSR_1 will be the input of LSR_2. Simultaneously, the same bit will inter to (wait-box3).
10. So on, the same procedure will take place for other steps until the competition between two bits in (wait-box8), which settled by bit controller8. If the bit of controller8 is (0) then the bit comes from (wait-box7) will be the input of LSR_8, but If the bit of controller8 is (1) then the bit comes from (LSR_7) will be the input of LSR_8.
11. From (step 10) the output bit yield from LSR_8 is the first output bit sequence of the proposed system.

2.3 System Evaluation

In this research, all requirements to build a robust system are taken. The proposal of a new generator (Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator) had been carefully designed to maximize security, so the first task to achieve this goal was to explore how to implement robust systems that do exhibit nonlinearity. As mentioned in [Section 2-1-2 Part two (Serial part; Suggested Part)], a new nonlinear technique was employed, to gain pseudorandom generator.

To evaluate the proposed algorithm, the complexity measures have been reviewed and their reaction to certain sequences is discussed. The complexity measurement is very important to evaluate binary key-stream for use in stream cipher cryptosystem to determine the size of a specific tool required in producing the output sequence; i.e. Turing Complexity (time complexity), Linear Complexity, the Maximum Order Complexity (Period), and Correlation Immunity. Also the measures related to patterns that appear in the sequence by the five statistical tests were used.

2.3.1 Time Complexity Measurements

The common notion for evaluating the algorithm is the concept of the order of magnitude of time complexity, where its expression Eq (2) is:

$$T(n) = (F(n)) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Eq(2)}$$

Where T(n) is the time algorithm on n inputs. The problem like the one in hand that cannot be solved in polynomial time is called intractable because calculating its solution quickly becomes infeasible, which is called hard. Alan Turing proved that such problems are undecidable (i.e. it's impossible to devise any algorithm to solve them).

As in literature,[3, 10] the definition of such undecidable problem, the satisfiability problem, is NP-Complete. The satisfiability problem is to determine if a formula is true for some assignment of truth-values to the variables.

So, the measurement to evaluate the (LFSRs Time Complexity) of this algorithm was done through analyzing the computing time, which is performed by execution of its statements given various sets of data. It is proven that for a large length LFSR the probability of severe degeneracy happening is virtually zero. In LFSRs implemented in this system the shift register is a sequence of bits, which is called an [(n) bits LFSR] when it consists of n bits long, then the complexity of a brute-force attack is $O(2^n)$. Herein there are two part of eight SRs for each part of them as below:

1. The eight independent LFSRs of [Part one (Traditional Part)], so their time complexity is the product of the complexity of all LFSRs, computed as in Eq (3):

$$T(n_1) = O(2^{39+61+47+25+70+100+35+127}) \qquad \text{……………Eq (3)}$$
$$= O(2^{504})$$

2. The eight LSRs of [Part two (Serial part; Suggested Part)], so their time complexity is the product of the complexity of all LSRs, computed as in Eq (4) below:

$$T(n2) = O(2^{91+88+96+84+98+94+87+100}) \qquad \text{…………..Eq (4)}$$
$$= O(2^{734})$$

So the total Time Complexity for both parts of the proposal is computed as in Eq (5) below:

$$T(n) = O(2^{504}) + O(2^{734}) \qquad \text{……..Eq (5)}$$
$$= O(2^{1238})$$

In addition to other time complexity, as space complexity, different functions and procedures had been used here in the algorithm of Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator.

2.3.2 Statistical Tests Measurement

In order to analyze such systems and configuration, the testing had been made to verify their strength against attacking by the most cryptanalysis techniques.
To prove that the proposed generator "Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator" has good statistical properties as clearly verifiable state, the submission of the different key-sequences of length (100000 bits) to the five statistical tests was done as below:

1- FREQUENCY TEST

```
***********************************
PASS VALUE 0.269 WITH FREEDOM DEGREE “1” MUST BE
<= 3.84
***********************************
```

2- RUN TEST

```
***********************************
PASS VALUE T0 = 19.541 WITH FREEDOM DEGREE “16”
MUST BE      <= 26.012
PASS VALUE T1 = 10.590 WITH FREEDOM DEGREE “13”
MUST BE      <= 22.078
***********************************
```

3- POKER TEST

```
***********************************
PASS VALUE 2.852 WITH FREEDOM DEGREE “5” MUST BE
<= 11.1
***********************************
```

```
   4-    SERIAL TEST
************************************
PASS VALUE 3.756 WITH FREEDOM DEGREE "3" MUST    BE
<= 7.81
************************************

  5-  AUTO_CORRELATION TEST
************************************
SHIFT NO.  1 >--> PASS VALUE 0.437
SHIFT NO.  2 >--> PASS VALUE 0.081
SHIFT NO.  3 >--> PASS VALUE 0.204
SHIFT NO.  4 >--> PASS VALUE 0.003
SHIFT NO.  5 >--> PASS VALUE 1.673
SHIFT NO.  6 >--> PASS VALUE 1.665
SHIFT NO.  7 >--> PASS VALUE 0.048
SHIFT NO.  8 >--> PASS VALUE 0.818
SHIFT NO.  9 >--> PASS VALUE 0.161
SHIFT NO. 10 >--> PASS VALUE 0.219
WITH FREEDOM DEGREE "1"          MUST BE     <= 3.84
FILE   OF   BINARY   DATA   =   C:\Documents   and
Settings\user\Desktop\Ran_Gen\seq_b.txt
THE LENGTH OF FILE = 100000 BITS
************************************

 1 - FREQUENCY T.
FRQ OF " 0 " = 50082     FRQ OF " 1 " = 49918
************************************
 2 - RUN T.
FRQ OF 1 " 0 " = 12502   FRQ OF 1 " 1 " = 12581
FRQ OF 2 " 0 " = 6280   FRQ OF 2 " 1 " = 6314
FRQ OF 3 " 0 " = 3137   FRQ OF 3 " 1 " = 3059
FRQ OF 4 " 0 " = 1575   FRQ OF 4 " 1 " = 1521
FRQ OF 5 " 0 " =   783   FRQ OF 5 " 1 " =   791
FRQ OF 6 " 0 " =   419   FRQ OF 6 " 1 " =   396
FRQ OF 7 " 0 " =   171   FRQ OF 7 " 1 " =   196
FRQ OF 8 " 0 " =    86   FRQ OF 8 " 1 " =    86
FRQ OF 9 " 0 " =    54   FRQ OF 9 " 1 " =    59
FRQ OF 10 " 0 " =    25   FRQ OF 10 " 1 " =    30
FRQ OF 11 " 0 " =    10   FRQ OF 11 " 1 " =     8
FRQ OF 12 " 0 " =     5   FRQ OF 12 " 1 " =     6
FRQ OF 13 " 0 " =     2   FRQ OF 13 " 1 " =     4
FRQ OF 14 " 0 " =     1   FRQ OF 14 " 1 " =     1
FRQ OF 15 " 0 " =     0   FRQ OF 15 " 1 " =     0
FRQ OF 16 " 0 " =     2   FRQ OF 16 " 1 " =     0
FRQ OF 17 " 0 " =     1   FRQ OF 17 " 1 " =     0
***********************************
 3 - POKER T.
 FRQ OF 0 " 1 " =   619
 FRQ OF 1 " 1 " = 3114
 FRQ OF 2 " 1 " = 6333
 FRQ OF 3 " 1 " = 6236
 FRQ OF 4 " 1 " = 3060
 FRQ OF 5 " 1 " =   638
***********************************
 4 - SERIAL T.
FRQ OF “00” = 12564     FRQ OF “11” = 12482
FRQ OF “01” = 12621     FRQ OF “10” = 12333
***********************************
 5 - AUTO_CORR. T.
 SHIFT NO.  1 FRQ OF “0” = 49895 FRQ OF “1” = 50104
 SHIFT NO.  2 FRQ OF “0” = 49954 FRQ OF “1” = 50044
 SHIFT NO.  3 FRQ OF “0” = 50070 FRQ OF “1” = 49927
 SHIFT NO.  4 FRQ OF “0” = 49989 FRQ OF “1” = 50007
 SHIFT NO.  5 FRQ OF “0” = 49793 FRQ OF “1” = 50202
 SHIFT NO.  6 FRQ OF “0” = 49793 FRQ OF “1” = 50201
 SHIFT NO.  7 FRQ OF “0” = 49962 FRQ OF “1” = 50031
 SHIFT NO.  8 FRQ OF “0” = 50139 FRQ OF “1” = 49853
 SHIFT NO.  9 FRQ OF “0” = 50059 FRQ OF “1” = 49932
 SHIFT NO. 10 FRQ OF “0” = 50069 FRQ OF “1” = 49921
```

So practically, the statistical tests were run for different key-sequences and all of them passed the practical tests successfully.

2.3.3 Correlation Immunity:

As shown in table (2) below, By virtue of the independent statistical distribution of the eight LFSRs sequences in a proposed generator, each sequence was individually tested with the output sequence of the final output generator affected by the virtue of the effect of the nonlinearity. The nonlinearity of both EPROMs and the function of suggested serial part have been considered to have maximal degree of immunity and to avoid the trade-off between linear complexity and correlation immunity. To implement Correlation Immunity Test, (6) different sequences used to imply results (MUST BE ≈ 0.5) as Table (2), which leads to say that the proposed " Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator " is of (8th) correlation immune. Therefore, that means the suggested algorithm is computationally secure against correlation attack.

Table (2) 8th correlation immune

Sequence 1	0.49883 0.50150	0.50004 0.49755	0.49979 0.49931	0.49931 0.49911
Sequence 2	0.49984 0.49956	0.50106 0.49924	0.49900 0.50061	0.49961 0.50080
Sequence 3	0.49936 0.50080	0.50106 0.49883	0.49853 0.50126	0.49717 0.49579
Sequence 4	0.50092 0.49679	0.49786 0.49847	0.49853 0.49940	0.49903 0.49851
Sequence 5	0.49836 0.50028	0.49925 0.49754	0.49912 0.50314	0.49871 0.49947
Sequence 6	0.49974 0.50041	0.49996 0.49925	0.50228 0.50112	0.49881 0.49962

2.3.4 Linear Complexity Measurement

Any sequence generated by a finite-state machine over a finite field has a finite linear complexity. Consequently, the linear complexity of a finite binary sequence s^n, denoted $L(s^n)$, is the length of the shortest LFSR that generates a sequence of the LFSR has s^n as its first n terms. That is, when the Sequence of (20000) Bits is practically tested, then the Length of LSR (Linear Equivalence) resulted is = 10000. So; the shortest length of LFSR that can generate equivalent sequence is (10000).
Therefore, the algorithm in this research is chosen so that the linear complexity (LC) resulted from the test of two sequence of 20000 bits that are generated from the ***Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator*** is enough to make the system computationally secure against linear attacks, as shown below:

```
****** Stream Cipher Results: Linear Complexity Test
****** File name: C:\Documents and Settings\user\Desktop\Ran_Gen\seq.txt
****** Time: Fr Jan 10 19:57:53 2014
Total bits                    : 20000
Linear complexity             : 10000
Expected linear complexity    : 10000
Significance probability (p)  : 0.5000
Number of jumps               : 5049
Expected number of jumps      : 5000
Significance probability (p)  : 0.8365
Jump size.
Chi-squared value             : 11.5626
Degrees of freedom            : 8
Significance probability (p)  : 0.1718
Interpretation.
Linear complexity test:
- 50.00% of bit streams of length 20000 will have a linear complexity less than this sample.
- This sample satisfies the linear complexity test.
```

Number of jumps:

- 83.65% of bit streams of length 20000 will have a number of jumps in linear complexity less than this sample.

- This sample satisfies the test on the number of jumps in linear complexity.

Jump Size:

- 17.18% of bit streams of length 20000 will have a jump size distribution further from the expected distribution than this sample gives.

This sample satisfies the test on the distribution of the linear complexity jump size.

2.4 Conclusions

In this research (***Implementing a New Serial Control as Nonlinear Function for Key Stream Pseudo-Random Number Generator***) an assertion of:

1. Cryptographically, obvious possibility of implementation of a new nonlinear part suggested for this generator.
2. The successful advantages of using the suggested part in this generator were clearly shown in the results obtained while the output sequence passed the randomness tests successfully in all practical experiments work.
3. Polynomial models offer fast linear parameter estimation with guaranteed solutions but simultaneously shown proportionally little flexibility for dealing with complex nonlinearities.
4. All above properties make the suggested generator suitable for practical applications.

References

[1] Adam Young & Moti Yung, “Malicious Cryptography Exposing Cryptovirology", Published in Canada, by Wiley Publishing, Inc., 2004.

[2] Beker H. & Piper F., "Cipher Systems: The Protection of Communication", Northwood Books, London, 1982.

[3] Bruce Schneier, "Applied Cryptography", Second Edition, New York Wiley, 1996.

[4] Elizabeth D. & Denning R., "Cryptography and Data Security", Addison-Wesley, 1983.

[5] Gustafson H., Dawson E., and Nielsen L., "A Computer Package for Measuring the Strength of Encryption Algorithms", Computers & Security, Elsevier Science Ltd, 1994.

[6] Huang and Andrew,"The key stream generator can be thought of as a cryptographic pseudo-random number generator (CPRNG)." No Starch Press. ISBN 9781593270292, 2003.

[7] James L. Massey, "Cryptography Fundamentals And Applications", Copies of Transparencies, James L. Massey, Copyright 1989.

[8] J.D. Golic, "Towards Fast Correlation Attacks on Irregularly Clocked SRs", Advances in Cryptology-EUROCRYPT , Proceedings 1995.

[9] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, "Cryptography Engineering: Design Principles and Practical Applications", Published by Wiley Publishing, March 8th 2010.

[10] Oded Goldreich, "Foundations of Cryptography", Publisher: Cambridge University Press, ISBN/ASIN: 0521035368 , 2007.

تطبيق السيطرةُ المتسلسلةُ كدالةٍ لاخطّيةٍ جديدةٍ لمولدِ المفتاحِ الانسيابي الرقميً شبه العشوائي

م. أحمد محمد مال الله الصفار
ahmed_malallah@yahoo.com
الجامعة العراقية - كليـــة الهندسة - قسم هندسة البرمجيات

المستخلص

أن السلوكُ اللاخطّيُ هو القاعدةَ وليس الإستثناءِ في الطبيعةِ، والتَقَدُّم في مجال الأمنِ والتشفير واسع جداً في العالمِ.

المشكلة : هناك حاجة مُلحة في التشفير للعمل الجاد لتقديم منتوج لاخطي أو تعقيد النظامِ الخطيِّ ليصبح نظاماً لاخطياً ويُعدُّ ذلك مسألة ذات إعتبار كبير.

بالتالي, لحل هذه المشكلة, تم أقتراح نظام لاخطي جديد لمولّدِ مفتاح رقمي شبه عشوائي في التشفير (تطبيق سيطرةَ متسلسلةَ جديدةَ كدالة لاخطّية لمولد المفتاح الانسيابي الرقمي شبه العشوائي) في هذا البحث, الذي يضم قسمين؛ الجزء التقليدي، كمجموعة من LFSRs، والجزء المسلسل كجزء مقترح وهو يضم نهجا جديدا بأداء عالي التعقيد لتلبية معايير التشفير القياسية.

عمليا، تم الحصول على مفتاح تشفير أنسيابي شبه عشوائي قوي في هذا البحث وهو يوفر زيادة كبيرة لِلّاخطية المعقدة، ويؤدي إلى أفضل تطوير للخصائص الاحصائية العشوائية للمتسلسلات الخارجة بمكافئ خطي كبير لمنع هجمات الارتباط, وقد إجتاز بنجاح جميع الاختبارات الإحصائية.

الكلمات الرئيسية: منتوج لاخطي، مفتاح تشفير، مسيطرات، جزء جديد لاخطي، مولد مفتاح انسيابي رقمي شبه عشوائي، نظام لاخطي، لوحة سيطرة البتات، الجزء المتسلسل.