Bulgarian Academy of Sciences. Space Research Institute.
Aerospace Research in Bulgaria. 21, 2007, Sofia

SOME STATISTICAL PROPERTIES OF SHRINKING–MULTIPLEXING GENERATOR

Zhaneta Tasheva

National Military University
e-mail: tashevi86@yahoo.com

Abstract

The binary additive stream ciphers are one of the cryptographic systems that are often used in aerospace and communication applications where high speed and low delay are requirements.

The need for software-oriented stream ciphers has lead to several alternative proposals in the last few years. A new Pseudo Random Number Generator (PRNG) named Shrinking–Multiplexing Generator (SMG) is described in this paper. It uses p-1 parallel working Linear Feedback Shift Registers (LFSRs) with large period and one p-adic Feedback with Carry Shift Registers (FCSR), which control nonlinearity in the generator.

To be suitable for use in cryptographic systems, PRNG must have large period, large linear complexity and good statistical properties. The results from statistical analysis of SMG are given in the paper. The sequence generated by SMG is uniform, scalable, uncompressible, with large period; consistent and unpredictable are shown by analysis of proposed PRNG. This gives us reason to say that the SMG is suitable for particular cryptographic application.

I. Introduction

The binary additive stream ciphers are one of the cryptographic systems that are often used in applications where high speed and low delay are requirements like modern aerospace and communication information systems.

In the binary additive stream cheaper, the keystream, the plaintext, and the ciphertext are sequences of binary digits. The keystream is generated by a keystream generator, which takes a secret key as a seed, and produces a long pseudorandom sequence. The ciphertext is generated by bitwise modulo 2 additions of the keystream and the plaintext.

The goal of stream cipher design is to efficiently produce pseudorandom sequences that in all senses are "indistinguishable" from truly random sequences. To do this, the pseudorandom sequence must have various properties, such as high linear span, high pair-wise Hamming distance, low cross-correlation value, large linear complexity and good statistical properties.

The need for software-oriented stream ciphers has lead to several alternative proposals in the last few years. To satisfy this need we propose a new Pseudo Random Number Generator (*PRNG*) and give its statistical analysis.

The paper is organized as follows. A new Shrinking–Multiplexing Generator (*SMG*) is described in Section II. The statistical analysis of the proposed *SMG* is given in Section III. Finally, some future works and conclusions are presented.

II. The Shrinking–Multiplexing Generator

In this section we give a brief review of the architecture and mathematical background of the new Shrinking–Multiplexing Generator (*SMG*) proposed by us in [8].

We generalized the architecture of the Shrinking generator [1] by using a p–adic control *FCSR R* (Fig. 1) instead of the control *LFSR* R_1. Thise increases a number of used slave *LFSRs* from 1 in Shrinking generator to $p-1$ in *SMG*. The control register *LFSR* R_1 is used to select a portion of the output sequences of a used *LFSRs* $R_1 \div R_{p-1}$.

Definition 1: A *Shrinking–Multiplexing Generator* (*SMG*) (Fig. 1) consists of a control *FCSR* of length L and of $p-1$ slaved *LFSRs* of length L_1, L_2, ..., L_{p-1}, each capable to produce one bit in its output; and a clock which controls the movement of data in registers. The algorithm of *SMG* with control p-adic *FCSR* consists of the following steps:

1. All *LFSRs* from R_1 to R_{p-1} and *FCSR R* are clocked.

2. If the p-adic output $b_1 = j$ of control register R is not equal to 0, the output bit of register R_j forms part of the keystream.

3. Otherwise, if the output $b_i = 0$ of control register R is equal to 0, the output bits of all slave registers R_1 to R_{p-1} discards.

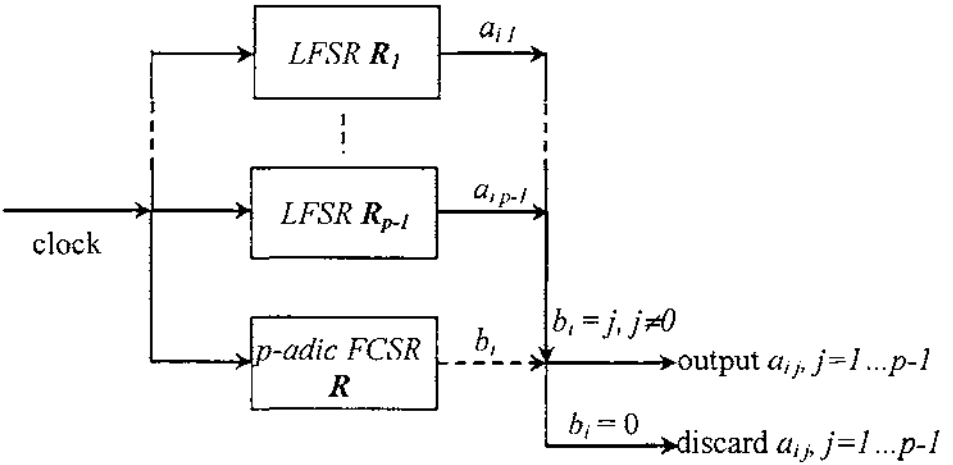

Fig 1. The Shrinking–Multiplexing Generator

Therefore, the produced keystream is a *shrunken* version of the output sequences of *LFSRs* R_1 to R_{p-1} when there is zero output of control *FCSR*, and a *mixed* version of the output sequences of *LFSRs* R_1 to R_{p-1} when output is not equal to zero.

The proposed *SMG* uses the generalization of 2-adic *FCSRs* [3] with stage contents and feedback coefficients in $\mathbf{Z}/(p)$ where p is a prime number, not necessarily 2.

Definition 2: A *p-adic feedback with carry shift register* (*FCSR*) with Galois architecture of length L (Fig. 3) consists of L *stages* (or *delay elements*) numbered *0, 1, ..., L-1*, each capable to store one p-adic (0, 1, ..., p-1) number and having one input and one output; and a clock which controls the movement of data. During each clock cycle the following operations are performed:

1. The content of stage 0 is output and forms part of the *output sequence*;
2. The sum modulo p after stage i is passed to stage i - *1* for each i, $1 \leq i \leq L-1$;

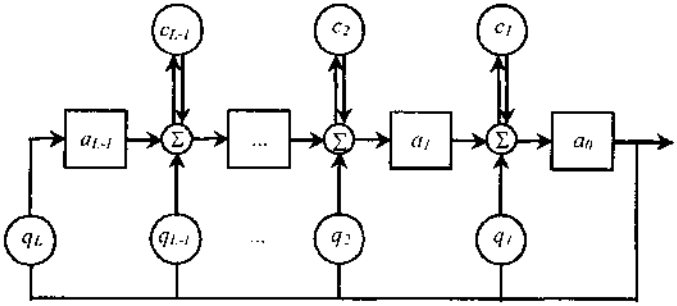

Fig. 2. Galois FCSR

3. The output of the last stage 0 is introduced into each of the tapped cells simultaneously, where it is fully added (with carry) to the contents of the preceding stages. The $q_1, q_2, \ldots, q_L$ are the *feedback multipliers* and the cells denoted by $c_1, c_2, \ldots, c_{L-1}$ are the *memory* (or "*carry*") bits. If

(1) $$q = -1 + q_1 p + q_2 p^2 + \ldots + q_L p^L$$

is the base p expansion of a positive integer:

(2) $$q \equiv -1 \pmod{p},$$

then q is a connection integer for a *FCSR* with feedback coefficients $q_1, q_2, \ldots, q_L$ in $\mathbf{Z}/(p)$.

With each clock cycle, the integer sums:

(3) $$\sigma_j = a_j + a_0 q_j + c_j$$

is accumulated.

At the next clock cycle this sum modulo p

(4) $$a'_{j-1} = \sigma_n \pmod{p}$$

is passed on to the next stage in the register, and the new memory values are:

(5) $$c'_j = \sigma_n \ (\mathrm{div}\, p).$$

The proposed *SMG* uses the *LFSRs* with stage contents and feedback coefficients in Extended Galois Field $GF(p^r)$. There are two architectures for *LFSRs*, the Galois and Fibonacci. We use the Galois one and next we give its definition [3].

Definition 3: A *linear feedback shift register* (*LFSR*) with Galois architecture of length L (Fig. 2) consists of L *stages* (or *delay elements*) numbered *0, 1, ..., L- 1*, each capable to store one bit and having one input and one output; and a clock which controls the movement of data. During each clock cycle the following operations are performed:

1. The content of stage 0 is output and forms part of the *output sequence*;
2. The content of stage i is moved to stage i - *1* for each i, $1 \le i \le L"1$;
3. The output of the last stage 0 is introduced into each of the tapped cells simultaneously, where it is added (modulo 2) to the contents of the preceding stages. If $q_1, q_2, ..., q_L$ are the *feedback multipliers* then the recurrence equations are as follows:

$$(6) \qquad a'_i = a_{i+1} + q_{i+1} a_0, \quad for\ 0 \le i \le L-2$$

$$a'_{L-1} = q_L a_0$$

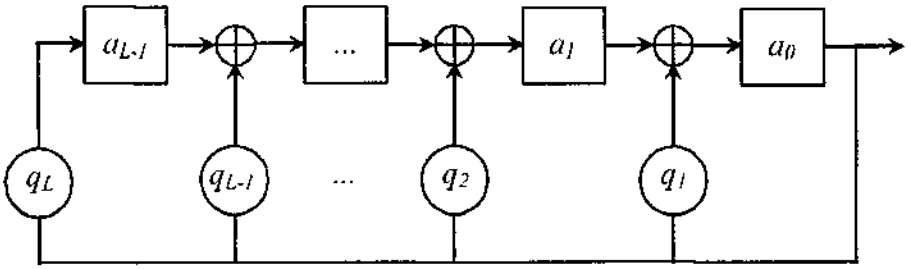

Fig. 3. Galois LFSR

Suppose a Galois *LFSR* with connection polynomial *q(X)*

$$(7) \qquad q(X) = -1 + \sum_{i=1}^{L} q_i X^i$$

has initial loading $(a_0, a_1, ..., a_{L"1})$. Then the output sequence $\boldsymbol{b} = (b_0, b_1, ...)$ of the *LFSR* is the coefficient sequence of the function

$$(8) \qquad B(X) = -\frac{h(X)}{q(X)},$$

where

(9) $h(x) = a_0 + a_1 X + \ldots + a_{L-1} X^{L-1}$.

Conversely, if $\boldsymbol{b}$ is any strictly periodic sequence and let $B(X) = -\dfrac{h(X)}{q(X)}$ be its generating function, then $q(X)$ is a connection polynomial of a Galois *LFSR* which generates the sequence $\boldsymbol{b}$, and $h(X)$ determines the initial loading by (4).

III. Statistical Properties of Shrinking–Multiplexing Generator

Generators suitable for use in cryptographic applications need to meet stronger requirements than for other applications. In particular, their outputs must be unpredictable, i.e. random. To test the statistical properties of *SMG* we have used some procedures proposed by the National Institute of Standards and Technology (*NIST*) to be useful in detecting deviations of a binary sequence from randomness. These tests show as a first step whether or not a generator is suitable for a particular cryptographic application.

The *NIST* statistical tests [5, 6] are formulated to test a specific null hypothesis (H_0): "The sequence being tested is random". Associated with this null hypothesis the alternative hypothesis (H_a) is that the sequence is not random. For each applied test a decision is derived that accepts or rejects the null hypothesis, i.e., whether the generator is (or is not) producing random values, based on the sequence that was produced.

For each test a statistical *P-value* is computed, which is a function of the data. For these NIST tests, each *P-value* [5] is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. If a *P-value* for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A *P-value* of zero indicates that the sequence appears to be completely nonrandom. A significance level (α) can be chosen for the tests. If *P-value* $\geq \alpha$, then the null hypothesis is accepted; i.e., the sequence appears to be random. If *P-value* $< \alpha$, then the null hypothesis is rejected; i.e., the sequence appears to be non-random. The parameter α denotes the probability of the Type I error. Typically, α is chosen in the range [0.001, 0.01]. In our test we chose $\alpha = 0.01$. This indicates that one would

expect 1 sequence in 100 sequences to be rejected. A $P\text{-}value \geq 0.01$ means that the sequence would be considered to be random with a confidence of 99 %. A $P\text{-}value < 0.01$ means that the conclusion was that the sequence is non-random with a confidence of 99 %.

Two types of errors are supposed by statistical hypothesis testing (Table 1. [5]). The conclusion, Type I error, occurs when the data is, in truth, random, and the test rejects the null hypothesis H_0 (the data is random). The Type II error occurs if the data is, in truth, non-random, and a conclusion to accept the null hypothesis H_0 is made.

Table 1. Possible statistical test outcomes

TRUE SITUATION	**CONCLUSION**	
	Accept H_0	***Accept Ha (reject H_0)***
Data is random (H_0 is true)	No error	Type I error
Data is not random (H_a is true)	Type II error	No error

To determine the randomness of arbitrarily long binary sequences produced by *SMG* we used the *NIST* Test Suite statistical package. It consists of 16 functional tests. Some of them are decomposable into a variety of subtests. Here we describe briefly the purpose of the tests and the characteristics that they detect. More information about the mathematical background of the tests the reader may be found in [2, 5, 6]. The 16 tests are:

1. ***The Frequency (Monobit) Test.*** The purpose of this test is to determine whether the number of ones and zeros in a sequence is approximately the same as would be expected for a truly random sequence. It detects too many zeroes or ones in the sequence.

2. ***Frequency Test within a Block.*** The purpose of this test is to determine whether the frequency of ones in an M-bit block is approximately $M/2$, as would be expected under an assumption of randomness. The test detects too many zeroes or ones at the beginning of the sequence.

3. ***The Runs Test.*** The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. A run of length k consists of exactly k identical bits and is bound before and after by a bit of the opposite value. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

4. ***Test for the Longest-Run-of-Ones in a Block.*** It determines whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. The small *P-values* indicate that the tested sequence has clusters of ones.

5. ***The Binary Matrix Rank Test.*** Test checks for linear dependence among fixed length substrings of the original sequence. The small *P-values* indicate a deviation of the rank distribution from that corresponding to a random sequence.

6. ***The Discrete Fourier Transform (Spectral) Test.*** The focus of this test is the peak heights in the Discrete Fourier Transform of the sequence. The test detects periodic features in the tested sequence.

7. ***The Non-Overlapping Template Matching Test.*** The purpose of this test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. If the *P-value* is very small (< 0.01), then the sequence has irregular occurrences of the possible template patterns. The test consists of 148 subtests for different aperiodic patterns.

8. ***The Overlapping Template Matching Test.*** Unlike the Non-Overlapping Template Matching Test, the focus of this test is the number of occurrences of prespecified target strings.

9. ***Maurer's "Universal Statistical" Test.*** The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random.

10. ***The Lempel-Ziv Compression Test.*** The test determines how far the tested sequence can be compressed, using the number of cumulatively distinct patterns (words) in the sequence. The sequence is considered to be non-random if it can be significantly compressed.

11. ***The Linear Complexity Test.*** The purpose of this test is to determine whether or not the sequence can be produced by some *LFSR*. A *LFSR* that is too short implies non-randomness.

12. ***The Serial Test.*** The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern.

13. ***The Approximate Entropy Test.*** This test compares the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence. It detects non-uniform distribution of m-length words.

14. ***The Cumulative Sums (Cusums) Test.*** It calculates the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. For a random sequence, the excursions of the random walk should be near zero. For certain types of non-random sequences, the excursions of this random walk from zero will be large. Test consists of two subtests.

15. ***The Random Excursions Test.*** The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The purpose of this test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence. This test is actually a series of eight tests (and conclusions), one for each of the states: -4, -3, -2, -1 and +1, +2, +3, +4.

16. ***The Random Excursions Variant Test.*** The test calculates total number of times that a particular state is visited in a cumulative sum random walk. It is actually a series of eighteen tests, one for each of the states: -9, -8, ..., -1 and +1, +2, ..., +9.

We test the *SMG* with 3-adic control register *FCSR* and 2 slave *LFSRs*. We test 3 different *SMGs* with connection tabs given on Table 2. For each generator we make 100 different tests with sequence of 1 000 000 bits, which is generated with a variety of seeds (different initial states). In all, this resulted in 300.189 = 56 700 *P-values*.

The results from all 56 700 tests are given on Table 3. On the table are given the average values for 100 *P-values* and 100 proportions for the 3 different *SMG* are tested. The number of tests is the same as those reviewed above. The letter "A" after the number of test indicates that the row gives the average value from all subtests included in this statistical test. The values of the cumulative-sums test (number 3.A) are the average from two subtests. The given values from the random-excursions test (number 12.A) are the average

from the eight subtests, the values from the random-excursions-variant test (number 13.A) are the average from the eighteen subtests, the values from nonperiodic-templates are the average (number 11.A) from 148 subtests, and the serial test (number 14.A) are the average from two subtests.

Table 2. Connections for tested SMG

№	*SMG*	Connection Integer *q*	Connection Polynomialsfor *R1* and *R2*
1.	*3-adic FCSR, L = 16;* *LFSR R_1, L1=131;* *LFSR R_2, L2=166.*	100 000 073	$x^{131} + x^8 + x^3 + x^2$; $x^{166} + x^{10} + x^3 + x^2$
2.	3-adic *FCSR, L = 16;* *LFSR R_1, L1 = 176;* *LFSR R_2, L2 = 200.*	100 000 037	$X^{176} + x^7 + x^5 + x^4 + x^3 + x^2 + 1$; $X^{200} + x^5 + x^3 + x^2 + 1$
3.	3-adic *FCSR, L = 14;* *LFSR R_1, L1 = 34;* *LFSR R_2, L2 = 32.*	10 000 229	$x^{34} + x^7 + x^6 + x^5 + x^2 + x + 1$; $x^{32} + x^7 + x^5 + x^3 + x^2 + 1$

To determine how well the empirical results match their theoretical counterparts we assess the goodness of fit of the distribution of *P-values* to a uniform distribution.

We build the histogram of *P-values* (fig. 4) among sub-intervals determined by dividing the unit interval by ten. To make a histogram all 567 average *P-values* from all statistical tests are used. As one can see from Fig. 4, the *P-values* are distributed approximately uniformly in all ten subintervals.

Table 3. Results from statistical analysis of SMG

№ of Test	Average from 100 tests of 1st *SMG*		Average from 100 tests of 2nd *SMG*		Average from 100 tests of 3th *SMG*		Average	
	P-value	Proportion	P-value	Proportion	P-value	Proportion	P-value	Proportion
1.	0,935716	1,00000	0,026948	1,0000	0,554420	0,9700	**0,505695**	**0,990000**
2.	0,236810	1,00000	0,437274	0,9900	0,129620	1,0000	**0,267901**	**0,996667**
3.A	0,357852	0,98500	0,505598	0,9900	0,586419	0,9750	**0,483290**	**0,983333**
4.	0,935716	0,99000	0,249284	0,9800	0,162606	0,9800	**0,449202**	**0,983333**
5.	0,834308	0,98000	0,867692	0,9900	0,699313	0,9700	**0,800438**	**0,980000**
6.	0,085587	1,00000	0,249284	0,9900	0,055361	1,0000	**0,130077**	**0,996667**
7.	0,779188	0,99000	0,032923	1,0000	0,935716	1,0000	**0,582609**	**0,996667**
8.	0,304126	1,00000	0,955835	0,9800	0,816537	0,9900	**0,692166**	**0,990000**
9.	0,964295	0,96000	0,162606	1,0000	0,236810	0,9600	**0,454570**	**0,973333**
10.	0,759756	0,99000	0,514124	0,9900	0,042808	0,9900	**0,438896**	**0,990000**
11.A	0,480569	0,99162	0,503849	0,9895	0,526539	0,9897	**0,503652**	**0,990292**
12.A	0,389515	0,97759	0,419667	0,9881	0,552950	0,9839	**0,454044**	**0,983183**
13.A	0,276655	0,99586	0,502274	0,9921	0,469981	0,9866	**0,416304**	**0,991487**
14.A	0,679079	1,00000	0,513301	1,0000	0,726828	0,9900	**0,639736**	**0,996667**
15.	0,657933	0,99000	0,075719	1,0000	0,401199	0,9800	**0,378284**	**0,990000**
16.	0,262249	0,97000	0,834308	0,9700	0,935716	1,0000	**0,677424**	**0,980000**

Then, the Goodness-of-Fit Distributional Test is made on the *P-values* obtained for an arbitrary statistical test using a distribution of χ^2. This is accomplished by computing:

$$(10) \qquad \chi^2 = \sum_{i=1}^{10} \frac{\left(F_i - \frac{s}{10}\right)^2}{\frac{s}{10}},$$

where F_i is the number of *P-values* in subinterval i and s is the sample size.

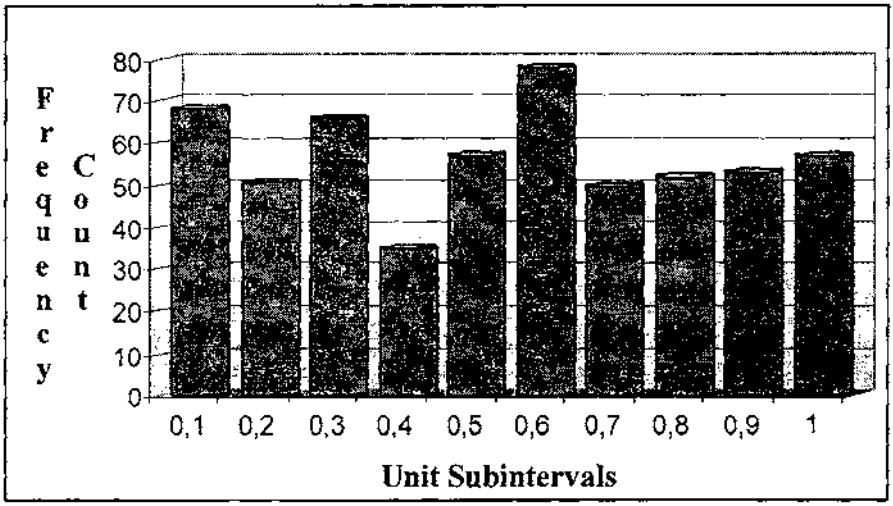

Fig. 4. Histogram of P-values

Thenthe *P-value* of the *P-values* is computed:

$$(11) \qquad P\text{-}value = Q\left(\frac{9}{2}, \frac{\chi^2}{2}\right),$$

where $Q(a,x)$is an Incomplete Gamma Function:

$$(12) \qquad Q(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^{\infty} e^{-t} t^{a-1} dt,$$

where $\Gamma(a)$ is a Gamma Function:

$$(13) \qquad \Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt.$$

Using the histogram of all 567 P-values in Fig. 4 and equations (10) – (13) *P–value = Q(4.5, 11.19) = 0.0055* are found. Because *P-value* ≥ 0.0001 [5], the sequence generated by *SMG* can be considered *uniformly distributed.*

To determine the proportion of sequences passing a statistical test we compute the proportion of sequences that had *P-values* $\geq$ *0.01*. The range of acceptable proportions is determined using the confidence interval defined as:

$$(14) \qquad \hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}},$$

where $\hat{p} = 1 - \alpha$, and m is the sample size. If the proportion falls outside this interval, then there is evidence that the data is nonrandom. Note that with m = 300 the confidence interval is 0.9728 ÷ 1.0. These bounds are given in Fig. 5 as dashed lines. As one can see from the figure, none of the proportions falls outside these thresholds.

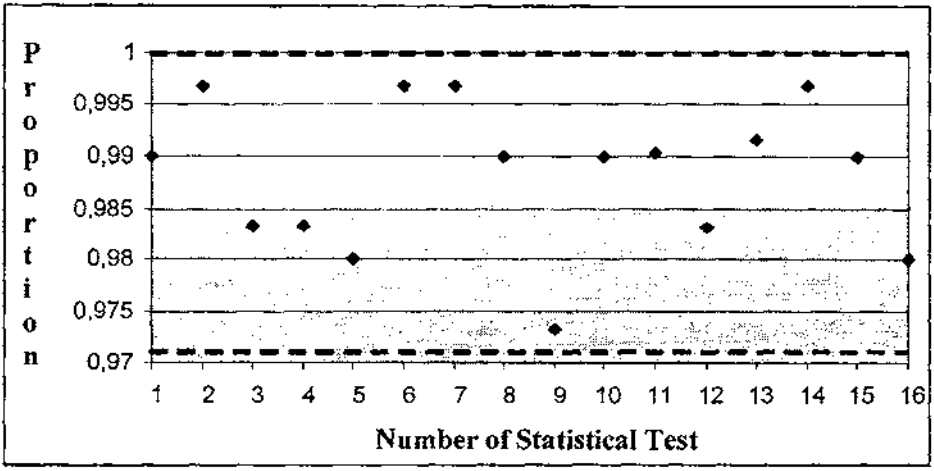

Fig. 5. Proportion of sequences passing a statistical test

The analysis of the empirical results from all 16 *NIST* statistical test shows:

1. The sequences generated by *SMG* are ***uniform***. This means that the probability of occurrence of a zero or one is equal, i.e. exactly 1/2. The expected number of zeros (or ones) is *n/2*, where *n* = the sequence length. This property is proved by follow tests and results:

1.1. The number of ones and zeros in a sequence and in sub-blocks are

approximately the same as in the truly random sequence is shown by 1st and 2nd *NIST* statistical tests. The average numbers for all 300 tested sequences are shown on Table 4.

Table 4. Number of Zeroes and Ones in SMG sequences

***SMG* Number**	**Average from 100 x 1 000 000 sequences**	
	Number of Zeroes	**Number of Ones**
SMG 1	499952	500048
SMG 2	500017	499983
SMG 3	500089	499911
Average from all *GSM*	**500019**	**499981**

1.2. The number of runs of ones and zeros of various lengths is as expected for a random sequence which is proved by 3rd test.

1.3. The results from 4th and 5th tests demonstrate that the generated sequences have not clusters of ones and there is not linear dependence among fixed length substrings of the original sequence.

1.4. The serial test (number 12) shows that every m-bit pattern has the same chance of appearing as every other m-bit pattern.

1.5. The excursions of the random walk is near zero which is given by test 14.

2. The sequences generated by *SMG* are ***scalable***, i.e. any extracted subsequence of sequence generated by *SMG* pass any test for randomness. This property is demonstrated by follow tests and results:

2.1. The approximate entropy test did not detect non-uniform distribution of m-length words.

2.2. The generated sequences have not irregular occurrences of the possible aperiodic template patterns which is shown by tests 7 and 8.

2.3. The number of visits to a particular state within a cycle did not deviate from the number expected for a random sequence.

3. The sequences generated by *SMG* are ***uncompressible***. This property is argued by follow tests and results which is by tests 15 and 16.

3.1. The sequence cannot be significantly compressed without loss of information which is shown by universal statistical test (number 9).

3.2. The sequence cannot be significantly compressed with Lempel-Zip algorithm (test 10).

4. The sequences generated by *SMG* have a ***large period***. This property is shown by follow results:

4.1. The spectral tests (number 9) did not detect periodic features in the tested sequence.

4.2. The linear complexity test (number 11) proved that the generated sequences cannot be produced by some too short *LFSRs* with length up to 16.

IV. Future works and conclusions

The empirical results from the above mentioned *NIST* statistical tests gives us reason to say that the binary sequences generated by *SMG* pseudorandom number generator have properties like ***uniformity, scalability, large period, incompressibility*** and ***consistency***, i.e. the behavior of a *SMG* is consistent across seeds. These properties show that the Shrinking–Multiplexing Generator works ***unpredictably*** like a true random number generator.

The proposed *SMG* is characterized with fast software implementation. It can be used in cryptographic applications in modern aerospace and communication information systems where high speed and low delay are requirements.

The proposed idea of using a p-adic control *FCSR* in *PRNG* allows generalizing all well-known clock controlled generators like p-adic Combiner Generator [7], Summation-Shrinking Generator [9], Alternating-Shrinking Generator and so on.

References

1. Coppersmith D., H. Karwczayk, Y. Mansou. "The Shrinking Generator", Crypto'93, http://imailab-www.iis.u-tocio.ac.jp/limit/Papers/Crypto_Eurocrypt/HTML/PDF/C93/22.pdf.

2. Getsov P., G. Mardirossian, S. Stoyanov, J. Jekov, P. Panova. "A Posibility of Storm and Hall Prediction using Data about Atmospheric Ozone Variations", Proceedings of International conference on Recent Advances in Spase Technologies – RAST, Istambul, 2003, pp.295-298.

3. Goresky M., A. Klapper. "Fibonacci and Galois Representations of Feedback-With-Carry Shift Registers", IEEE Trans. Inform. Theory, vol. 48, pp. 2826"2836, November 2002.
4. Menezes A., P. Oorschot, S. Vanstone. Handbook of Applied Cryptography, CRC Press, p. 780, 1997, www.cacr.math.uwaterloo.ca/hac.
5. Rukhin A., J. Soto, J. Nechvatal, M. Smid, El. 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 800-22 (with revisions dated May 15, 2001), p. 162, 2001.
6. Soto J.. Statistical Testing of Random Number Generators, NIST Special Publication (301) 975-4641, p. 12, 2001.
7. Stoyanov B., B. Bedzhev. Algorithm for *p*-adic Combiner Generator Synthesis, XXXIX International Scientific Conference on Information, Communication and Energy Systems and Technology, ICEST 2004, 16-19 June 2004, Bitola, Macedonia, under printing.
8. Tasheva Zh., B. Bedzhev, V. Mutkov. "A Shrinking Data Encryption Algorithm with p-adic Feedback with Carry Shift Register", Conference Proceedings of XII International Symposium on Theoretical Electrical Engineering, ISTET'03, July 6-9, 2003, Warsaw, Poland, Volume II, pp. 397 – 400, 2003.
9. Tasheva Zh., B. Bedzhev, B. Stoyanov. Summation-Shrinking Generator, International Conference "Information Technology and Sequrity ITS – 2004", June 22-26, 2004, Partenit, Crimea, Ukraine, under printing.

НЯКОИ СТАТИСТИЧЕСКИ СВОЙСТВА НА СВИВАЩИЯ-МУЛТИПЛЕКСИРАЩ ГЕНЕРАТОР

Ж. Ташева

Резюме

Двоичните поточни шифри са едни от криптографските приложения, които често се използват в космическите и комуникационни системи, където се изискват високи скорости и малки закъснения.

Необходимостта от високоскоростни софтуерни поточни шифри през последните години доведе до предлагането на няколко нови алтернативни решения. В статията накратко е представена архитектурата

и принципа на работа на нов псевдослучаен генератор (*PRNG*), наречен Свиващ-Мултиплексиращ Генератор (Shrinking–Multiplexing Generator - *SMG*). Той е изграден от p - 1 паралелни линейни преместващи регистри с обратни връзки (*LFSRs*) с голям период и един преместващ регистър с пренос и обратни връзки (*FCSR*), който управлява нелинейността в работата на *SMG*.

За да бъде приложим в криптографските системи, всеки *PRNG* трябва да има големи период и линейна комплексност, както и добри статистически свойства. В статията са представени резултатите от направения статистически анализ на 300 различни последователности с дължина 1 000 000 бита, генерирани при различни ключове на *SMG* генератора. За анализа е използван предложения от Националния институт за стандарти и технологии комплект от 16 различни функционални статистически тестове. Анализът потвърждава, че генерираните от *SMG* последователности са еднообразни, независещи от използваното ядро на генератора, с голям период и не могат да се компресират. Всички тези свойства се отнасят и за всички подредици на генерираната последователност. Тези резултати ни дават основание да твърдим, че генерираната от *SMG* последователност е непредсказуема и се характеризира със свойствата на истински случаен генератор.