Self-Shrinking p-adic Cryptographic Generator

Zhaneta N. Tasheva [1], Borislav Y. Bedzhev [2], Borislav P. Stoyanov [3]

Abstract – **A new cryptographic pseudo random number generator (*PRNG*), called Self–Shrinking *p*–adic Generator (*SSPG*), is proposed in this paper. The *SSPG* sequence is evaluated and its balancing is proved. The results of statistical analysis of *SSPG* sequence are given. They show that the sequence generated by a *SSPG* is uniform, scalable, uncompressible, consistent, unpredictable and has large period. This gives the reason to consider the *SSPG* as suitable for a particular software cryptographic application in stream ciphers.**

Keywords – **Cryptography, stream cipher, PRNG.**

I. INTRODUCTION

The stream ciphers are an important tool for protecting information in digital form and for providing security services. The performance quality of the hardware and software stream ciphers depends on their crypto resistance, velocity and effectiveness. Mostly the crypto resistance of a stream cipher is connected with it ability to generate Pseudo Random Sequence with enormous period, uniform distribution of *d*-tuples for a large range of *d* and with good usually lattice structure in high dimensions.

In order to achieve high performance velocity and cost-effective software implementation, the Pseudo Random Number Generator (*PRNG*) architecture must be simple on the one hand and on the other must be combine with some nonlinear functions fast and cheap elements, like as Linear Feedback Shift Registers (*LFSRs*) and Feedback with Carry Shift Registers (*FCSRs*). Recently, some theorists [6], [8], [10] have used this new approach of stream cipher design and have proposed a few new architectures named Shrinking Generator [1] and Self–Shrinking Generator [5]. They are promising candidates for high-speed encryption applications due to their simplicity and provable properties. With regard, main goal of our paper is to suggest a novel Self–Shrinking Generator, utilizing *FCSRs*.

The paper is organized as follows. First, the basic theory of the self–shrinking generator is recalled. Second a new *PRNG* architecture, called Self–Shrinking p–adic Generator (*SSPG*) is presented. After then, some properties and statistical analysis of the *SSPG* sequence are given. Finally, the possible application areas of the *SSPG* are discussed.

[1]Zhaneta N. Tasheva is an Assistant Prof. Eng. PhD. in the Faculty of Artillery and Air Defence, National Military University, Shoumen, 1st Karel Shkorpil Str., Shoumen 9710, Bulgaria, E-mail: tashevi86@yahoo.com

[2]Borislav Y. Bedzhev is an Assoc. Prof. Eng. DSc. in the Faculty of Artillery and Air Defence, National Military University, Shoumen, 1st Karel Shkorpil Str., Shoumen 9710, Bulgaria, E-mail: bedzhev@mail.pv-ma.bg

[3]Borislav P. Stoyanov is an Assistant Prof. Mag. PhD. Student in the Shoumen University, Faculty of Computer Informatics, Shoumen, Bulgaria, E-mail: bpstoyanov@abv.bg

II. SELF-SHRINKING GENERATOR

Both the Shrinking Generator and Self-Shrinking Generator use the *LFSRs* and have a simple structure. Despite of this fact no successful cryptanalytic attack for both generators has been published so far.

The self-shrinking generator uses only one *LFSR* whose output sequence is shrunken under the control of the *LFSR* itself [5]. The self-shrinking can be applied to any binary sequences. In this process the original sequence $a=(a_0, a_1, a_2, \ldots)$ is considered as a sequence of pairs of bits. If a pair (a_{2i}, a_{2i+1}) equals the value $(1,0)$ or $(1,1)$, it is taken to produce the pseudo random bit 0 or 1, respectively. On the other hand, if the pair is equal to $(0,0)$ or $(0,1)$, it will be discarded, which means that it will not contribute an output bit to the new sequence $s=(s_0, s_1, s_2, \ldots)$.

Below some properties of self-shrunken maximum length *LFSR*-sequence will be recalled. The proofs of given theorems can be found in [5].

Theorem 1: The period *P* of a self-shrunken maximum length *LFSR*-sequence produced by an *LFSR* of length *N* satisfies:

$$P \geq 2^{\lfloor N/2 \rfloor}. \tag{1}$$

Theorem 2: The linear complexity *L* of a self-shrunken maximum length *LFSR*-sequence produced by an *LFSR* of length *N* satisfies:

$$L > 2^{\lfloor N/2 \rfloor - 1}. \tag{2}$$

The experimental results, shown by Willi Meier and Othmar Stafelbach [5], reveal that the linear complexity does not exceed the value $2^{N-1} - N + 2$.

At the end of this section, it ought to emphasize that the simple algebraic structure of the original *LFSR*-sequence has been destroyed during the self-shrinking due to the reasons:

- randomness of the positions, where the *LFSR*-sequence is shrunken;
- the *LFSR* is controlled by itself.

III. SELF-SHRINKING P-ADIC GENERATOR

In this section the basic architecture of a new Self-Shrinking *p*-adic Generator (*SSPG*) and some its properties will be presented.

A. The SSPG Architecture

The *SSPG* architecture (Fig. 1.) uses a *p*-adic *FCSR* [3] instead of a *LFSR* in contrast with the classic self-shrinking generator. This allows the generator to produce a number from 0 to $p-1$ in one step ($a_i = [0, 1, \ldots, p-1]$). The self-shrinking *p*-adic generator selects a portion of the output *p*-adic *FCSR* sequence controlling the *p*-adic *FCSR* itself by means of the following algorithm.

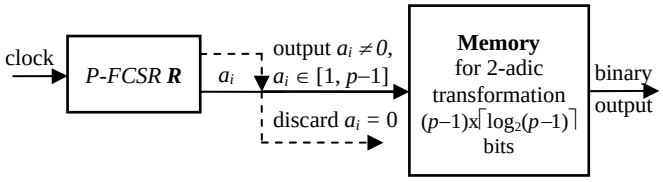

Fig. 1. Self-Shrinking *p*-adic Generator

Definition 1: The algorithm of the Self-Shrinking *p*-adic Generator (Fig. 1) consists of the following steps:

1. The *p*-adic *FCSR R* is clocked with clock sequence with period τ_0.

2. If the *p*-adic *FCSR* output number is not equal to 0 ($a_i \neq 0$), the output number forms a part of the *p*-adic *SSPG* sequence. Otherwise, if the output number of the *p*-adic *FCSR* is equal to 0 ($a_i = 0$), the *p*-adic output number of *SSPG* is discarded.

3. The shrunken *p*-adic *SSPG* output sequence is transformed in a usual binary sequence presenting every *p*-adic number with $\lceil \log_2 (p-1) \rceil$ binary digits, where $\lceil x \rceil$ is the smallest integer that is greater than or equal to real *x*. After that, every binary output number ***i***, ranging from **1** to ***p*–1**, is replaced with the binary number:

$$i - 1 + \frac{2^{\lceil \log_2 (p-1) \rceil} - (p-1)}{2}. \qquad (3)$$

TABLE I
BINARY PREZENTATION OF P-ADIC SSPG OUTPUT

***p*-adic number**	**Binary presentation of *p*-adic number**				
	***p* = 3**	***p* = 5**	***p* = 7**	***p* = 11**	***p* = 13**
1	0	00	001	0011	0010
2	1	01	010	0100	0011
3	–	10	011	0101	0100
4	–	11	100	0110	0101
5	–	–	101	0111	0110
6	–	–	110	1000	0111
7	–	–	–	1001	1000
8	–	–	–	1010	1001
9	–	–	–	1011	1010
10	–	–	–	1100	1011
11	–	–	–	–	1100
12	–	–	–	–	1101

The binary presentations of *p*-adic shrunken *SSPG* output numbers are shown with various prime *p* from 3 to 13 in Table I.

The proposed *SSPG* uses the generalization of 2-adic *FCSRs* [2], [3], [4] with stage contents and feedback coefficients in Z/(*p*) where *p* is a prime number, not necessarily 2.

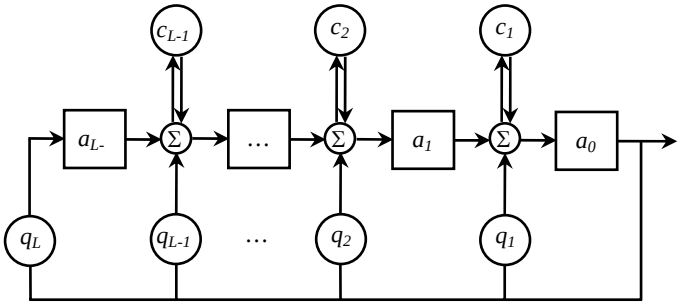

Fig. 2. Galois FCSR

Definition 2: A *p-adic feedback with carry shift register* (*FCSR*) with Galois architecture of length *L* (Fig. 2.) consists of *L stages* (or *delay element*s) 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 sequenc*e;

2. The sum modulo *p* after stage *i*, depicted as "∑" on Fig. 2, passes 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 fully (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 with $c_1, c_2, \ldots, c_{L-1}$ are the *memory* (or "*carry*") bits. If

$$q = -1 + q_1 p + q_2 p^2 + \ldots + q_L p^L \qquad (4)$$

is the base *p* expansion of a positive integer:

$$q \equiv -1 \pmod{p}, \qquad (5)$$

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

With each clock cycle, the integer sums:

$$\sigma_j = a_{j+} a_0 q_j + c_j \qquad (6)$$

is accumulated.

At the next clock cycle this sum modulo *p*

$$a'_{j-1} = \sigma_n \pmod{p} \qquad (7)$$

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

$$c'_j = \sigma_n \ (\text{div } p) . \qquad (8)$$

The nonlinearity of the proposed *SSPG* follows from the fact that it is unknown at which positions the *FCSR*-sequence is shrunken. As a result the linear algebraic structure of the original *FCSR*-sequence is destroyed. The software *SSPG* implementation is very fast because the pseudorandom generator produces $\lceil \log_2 (p-1) \rceil$ binary digits in every step.

B. The SSPG properties

In this subsection the period of *SSPG* sequences generated by maximum length *p*-adic *FCSR* will be established and it will be proved that the *SSPG* sequence is balanced.

Theorem 3: The period of the self-shrunken *p*-adic generator realized by maximum length *p*-adic *FCSR* of length *L* and connection integer *q* is:

$$T_{SSPG} = T^* . \lceil \log_2 (p-1) \rceil , \qquad (9)$$

where *T** is the quantity of output *p*-adic *FCSR* nonzero numbers.

Proof: Let $\boldsymbol{a} = (a_0, a_1, a_2, \ldots)$ be the output sequence of trivial *p*–adic *FCSR* (Fig. 2) of length *L* and connection integer *q* (Eq. (4)). By definition $\boldsymbol{a}$ is a maximum length sequence. Consequently, its period is *T*, where *T* is the multiplicative order of *p* modulo *q* (i.e. T is the smallest integer with property $p^T \equiv 1 (\text{mod } q)$) [10]. The self-shrunken *p*-adic sequence is periodic also, because every *SSPG* is a digital automat with limited quantity of possible inner states. Hence after appearing of all *T** nonzero elements of the original *FCSR* sequence, the output shrunken *p*-adic sequence will be repeated again. During the step 3 of *SSPG* algorithm every *p*-adic element of self-shrunken sequence is transformed into exactly $\lceil \log_2 (p-1) \rceil$ binary digits. Consequently the period of self-shrunken *SSPG* sequence is $T^* . \lceil \log_2 (p-1) \rceil$.

Theorem 4: The self-shrunken output *SSPG* sequence generated by maximum length *p*-adic *FCSR* of length *L* and connection integer *q* is a balanced sequence.

Proof: As is it known [2], [3], within the period of a *p*-adic *FCSR* sequence each of *p*–adic numbers from 0 to $p-1$ appears with approximately equal probability, i.e. every *p*–adic number, ranging from 1 to $p-1$, appears in the self-shrunken *p*–adic sequence approximately N_p times:

$$N_P \approx \left\lceil \frac{T}{p} \right\rceil . \qquad (10)$$

The *SSPG* algorithm utilizes a binary transformation of *p*–adic *FCSR* output elements during the step 3, which provides balanced distribution of binary digits 0 and 1. In order to prove this fact two cases will be considered.

Fist case: If the prime *p* can be present as $2^n + 1$, i.e. the odd number $p-1$ is a power of 2, then the output *p*–adic numbers from 1 to $p-1$ will be transformed into all binary numbers from 0 to $2^n - 1$ (may not be in a successive order). It is apparent that every permutation of the binary numbers from 0 to $2^n - 1$ is balanced, i.e. the number of 0s and 1s is exactly equal to $n.2^{n-1}$. This fact can be illustrated by means of Table I where:

- for $p = 3 = 2^1 + 1$ the number of 0s and 1s is equal to 1;
- for $p = 5 = 2^2 + 1$ the number of 0s and 1s is exactly $4 = 2.2^1$.

Consequently, if the prime *p* can be present in the form $p = 2^n + 1$, the numbers of 0s and 1s in the self-shrunken output *SSPG* sequence are balanced and equal to:

$$N_{0S} \approx N_{1S} \approx \left\lceil \frac{T}{2^n + 1} \right\rceil 2^{n-1} . n . \qquad (11)$$

Second case: If the odd number $p-1$ is smaller than $2^{\lceil \log_2 (p-1) \rceil}$:

$$p - 1 < 2^{\lceil \log_2 (p-1) \rceil} , \qquad (12)$$

then the smallest and the biggest $\frac{2^{\lceil \log_2 (p-1) \rceil} - (p-1)}{2}$ binary numbers in the range $0 \div 2^{\lceil \log_2 (p-1) \rceil}$ -1 of all possible binary numbers are rejected during the step 3. Hence the quantity of 1s and 0s is balanced also and equal to $\frac{\lceil \log_2 (p-1) \rceil . (p-1)}{2}$. This fact can be illustrated by means of Table I where:

- for $p = 7 = 2^3 - 1$ the number of 0s and 1s is equal to $9 = 3.3 = 3.(7-1)/2$;
- for $p = 11 = 2^4 - 5$ the number of 0s and 1s is equal to $20 = 4.5 = 4.(11-1)/2$;
- for $p = 13$ the number of 1s and 0s is $24 = 4.6$.

Consequently, if the prime *p* satisfies the inequality:

$$p < 2^{\lceil \log_2 (p-1) \rceil} + 1 \qquad (13)$$

the numbers of 0s and 1s in the self-shrunken output *SSPG* sequence are balanced and equal to:

$$N_{0S} \approx N_{1S} \approx \frac{1}{2} \left\lceil \frac{T}{2^n + 1} \right\rceil \lceil \log_2 (p-1) \rceil (p-1) . \qquad (14)$$

The transformation in step 3 (see Eq. (3)) eliminates the possibility of appearance the sequences of $2\lceil \log_2 (p-1) \rceil$ consecutive 1s and $2\lceil \log_2 (p-1) \rceil$ consecutive 0s also.

IV. Statistical Experimental Results

The randomness of binary sequences generated by *SSPG* was investigated by so-named "NIST suite", proposed by National Institute of Standards and Technology (USA). The NIST suite [7] includes sixteen tests. The tests examines on a variety of different types of non-randomness that could exist in a sequence. These tests are: frequency (monobit), frequency within a block, runs, longest-run-of-ones in a block, binary matrix rank, discrete Fourier transform (spectral), non-

overlapping template matching (consists of 148 subtests), overlapping template matching, Maurer's "Universal statistical", Lempel-Ziv compression, linear complexity, serial (consists of 2 subtests), approximate entropy, cumulative sums (consists of 2 subtests), random excursions (consists of 8 subtests), random excursions variant (consists of 18 subtests).

The testing process consists of the following steps [7], [9]:

1. State the null hypothesis. Assume that the binary sequence is random.

2. Compute a sequence test statistic. Testing is carried out at the bit level.

3. Compute the *p*–value, $p - \text{value} \in [0, 1]$.

4. Compare the *p*–value to error probability α. Fix α, where $\alpha \in (0.0001, 0.01]$. *Success* is declared whenever $p - \text{value} \geq \alpha$; otherwise, *failure* is declared.

The 1 000 sequences of length 1 000 000 bits, generated by *SSPG* with $p = 5$, are tested. The seed of *SSPG* are changed in every 1 000 bits by modifying the connection taps, initial state and initial memory state of a *5*–adic *FCSR*. The results from all NIST statistical tests are given in Table II.

TABLE II
THE RESULTS FROM 5–ADIC SSPG STATISTICAL TESTS

Statistical Tests	**Results**
Frequency (monobit)	Pass
Frequency within a block	Pass
Cumulative sums	Pass
	Pass
Runs	Pass
Longest-run-of-ones in a block	Pass
Binary matrix rank	Pass
Discrete Fourier transform (spectral)	Pass
Non-overlapping template matching	Pass
Overlapping template matching	142 Subtests Pass 6 Subtests Failure
Maurer's "Universal statistical"	Pass
Approximate entropy	Pass
Random excursions	All 8 Subtests Pass
Random excursions variant	All 18 Subtests Pass
Serial	Pass
	Pass
Lempel-Ziv compression	Pass
Linear complexity	Pass

As one can see from Table II, most of the NIST statistical tests are passed. Only 6 subtests of non-overlapping template test are failed. It was observed that the distributions of *p*–values of sequences, passed the statistical tests, aren't distributed uniformly, i.e. the numbers of *p*–values that lie within each unity sub-interval aren't equal.

V. CONCLUSION

In this paper the architecture of new self-shrinking *p*–adic generator is suggested. A few important properties of *SSPG* sequences generated by maximum length *p*-adic *FCSR* are established. The results from statistical analysis show that the sequence generated by *SSPG* is uniform, scalable, uncompressible, unpredictable and has large period. This gives the reason to consider the *SSPG* as a fast software pseudorandom generator and it can be useful as a part of modern stream ciphers.

ACKNOWLEDGEMENT

We will be glad to thanks everyone who helps us to make some strong cryptanalysis of self-shrinking *p*–adic generator.

REFERENCES

[1] D. Coppersmith, H. Krawczyk, Y. Mansour, "The Shrinking Generator", *Proceedings of Crypto 93*, Springer-Verlag, pp. 22-39, 1994.

[2] A. Klapper, M. Goresky, "2-adic Shift Register. Fast Software Encryption", *Second International Workshop*, (Lecture Notes in Computer Science, vol. 950, Springer Verlag, N. Y.,) pp. 174–178, 1994.

[3] A. Klapper, M. Goresky, "Feedback Shift Registers, 2-adic Span, and Combiners With Memory.", *Journal of Cryptology*, Volume 10, Number 2, 1997, pp. 111-147, http://www.math.ias.edu/~goresky/pdf/2adic.jour.pdf

[4] A. Klapper, J. Xu, "Algebraic Feedback Shift Registers" *Elsevier Preprint*, 2003.

[5] W. Meier, O. Staffelbach, "The Self-Shrinking Generator" *Proceedings of Advances in Cryptology*, *EuroCrypt '94*, Springer-Verlag, pp. 205-214, 1998.

[6] P. van Oorshot, A. Menezes, S. Vanstone, "*Handbook of Applied Cryptography"*, CRC Press, 1997.

[7] 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 Pseudo-Random Number Generators for Cryptographic Application", *NIST Special Publication 800-22* (with revision May 15, 2001) http://csrc.nist.gov/rng/.

[8] B. Schneier, "*Applied Cryptography"*, John Wiley & Sons, New York, 1996.

[9] J. Soto, "Statistical Testing of Random Number Generators", *NIST Special Publication*, http://csrc.nist.gov/rng/.

[10] Zh. N. Tasheva, B. Y. Bedzhev, V. A. Mutkov, "An Shrinking Data Encryption Algorithm with *p*-adic Feedback with Carry Shift Register", *Conference Proceedings of XII International Symposium of Theoretical Electrical Engineering ISTET 03*, Warsaw, Poland, 6-9 July, 2003., vol. II, pp. 397–400.