Increasing the quality of pseudo-random number generator based on fuzzy logic

I.V Anikin[1], K. Alnajjar[1]

[1]Information Security Systems Department, Kazan National Research Technical University named after A.N. Tupolev-KAI, Kazan, Russia, 420111

Abstract. We consider previously suggested by authors pseudorandom number generator based on fuzzy logic (FRNG) and increase its quality by increasing the period of generated series. To increase the period of FRNG we use more LFSRs in its structure. At the beginning we will study the case of combining four LFSRs in constructing the FRNG then the case of using eight LFSRs. The new version of FRNG generates pseudorandom numbers which are more close to true random series that make it more suitable and attractive for many applications related to data security and cryptography.

Keywords: pseudorandom number generator, random series, fuzzy logic.

1. Introduction

Nowadays, it is well known that pseudorandom number generation is a concept of fundamental importance in many different areas of computer science and digital communications. A good pseudorandom sequence should have a good statistical properties and should satisfy number of important requirements as unpredictability, nonlinearity, Golomb's three randomness postulates for binary sequences, immunity against correlation attacks, and it's period should be long enough since randomness is meaningless for short sequences. Due to high speed of operations, simplicity and very good statistical properties of the generated pseudorandom sequences, pseudorandom number generators (PRNGs) based on combining the outputs of LFSRs using a nonlinear function, are the most common long-period generators available at the present time [1].

The proposed generator FRNG based on fuzzy logic techniques [2] to create a new non-linear function that helps in combining a number of LFSRs as depicted in ‘figure 1’. Firstly the outputs of LFSRs pass through the buffers of the same size (32bits), where an estimation of two fuzzy statistical variables over the each buffer takes place. These two linguistic variables are f_0 (number of one in the buffer) and $|f_1-f_2|$ (the difference between the number of runs consisting of two ones f_1 and the number of runs consisting of two zeros f_2). Then using a group of fuzzy If-Then rules the proposed system evaluates the statistical state of every buffer at the considered moment in order to select the LFSR with the best statistical properties and select its output bit to be the output of the system at the moment, to be a bit of the pseudorandom series generated by FRNG then a new evaluation of the linguistic

variables (f_0, $|f_1-f_2|$) associated with every buffer begins (after shifting the contents of the buffers one bit to the right and inserting the output bit of the related LFSR as LSB) to select the next bit and pass it to the output of the system and so on. In [3, 4] we investigated the parameters of FRNG and proved that the suggested generator produces a pseudorandom series that has good statistical properties by testing them using the most powerful randomness tests packets (NIST, DIEHARD) [5].

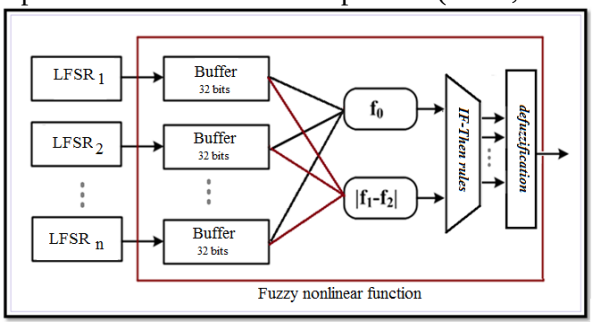

Figure 1. The construction of the suggested FRNG.

It's worth mentioning, that in our previous researches we used only two LFSRs in constructing the FRNG in order to simplify studying of its parameters.

In this paper we aim to introduce a new versions of FRNG that has high quality and more secure against algebraic attacks by increasing the period of generated series, via using more LFSRs that makes the generated series very close to true random streams. At the beginning we will study the case of combining four LFSRs in constructing the FRNG then the case of using eight LFSRs. So the new versions of FRNG will have larger periods and will produce pseudorandom sequences with high level of randomness that make it more suitable and attractive for many applications related to data security and cryptography.

In [3] we proved that the period of the generated sequence by FRNG is equal to the product of periods of used LFSRs, when a characteristic polynomials of LFSRs are selected as primitive polynomials. So the period of the output stream (T_s) is defined by following formula:

$$T_s = T_1 \cdot T_2 \cdots T_n = \left(2^{d_1}-1\right)\left(2^{d_2}-1\right)\cdots\left(2^{d_n}-1\right);$$

where $d_1, d_2, \cdots, d_n$ represent the degrees of characteristic primitive polynomials of used LFSRs respectively.

2. Selecting the characteristic primitive polynomials for FRNG

A good PRNG are required to have characteristic polynomials, whose number of nonzero coefficients is not too far from half of the degrees [5]. On other hand, selecting primitive polynomials with such a big number of terms leads to high hardware cost. So we should find a compromised solution. Very good solution to this problem suggested in [6], it guides us to a method for finding polynomials with minimum-cost in hardware implementation and with suitable number of terms. The suggested method can be summarized in finding a primitive polynomial, satisfying the following formula:

$$P(x) = \left(1+x^{b_1}\right)\left(1+x^{b_2}\right)...\left(1+x^{b_m}\right)+x^n \quad (1)$$

Then a minimum-cost LFSR that implements the same *P(x)* can be constructed using (*m*) *2-input XOR* gates, where the parameters of this type of polynomials $(b_1, b_2, ..., b_m, n)$ should satisfy the following conditions [6]:

$$b_1 \geq 1,\ b_1 < b_2, \left(b_1+b_2\right) < b_3, \cdots, \left(b_1+b_2+....+b_{m-1}\right) < b_m,\ \left(b_1+b_2+....+b_m\right) < n$$

In addition to the primitivity tests, that the selected polynomial should pass to be used in constructing the FRNG. It's worth to note, that number of non-zeros coefficients of these type of polynomials defined by ($t = 2^m + 1$), so with *m*=5 we will get polynomials with *t*=33 which is very good when the degree of polynomial is between 67 and 131. In the following paragraphs of this paper we will refer briefly to the selected characteristic primitive polynomials using their

parameters $(b_1, b_2, ..., b_m, n)$. For example, the tuple (1,4,7,20,53,97) refers to the following primitive polynomial:

$$P(x) = (1+x)(1+x^4)(1+x^7)(1+x^{20})(1+x^{53})+x^{97}$$

Due to the high Hamming weight (number of non-zero coefficients of the polynomial) and low power consumption (very few of XOR gates are needed in hardware implementation) using such type of polynomials will definitely improves the statistical properties and thus increases the security of the designed generator.

3. Constructing the FRNG using four LFSRs

Firstly we will construct our new FRNG using four LFSRs concerning the correlation immunity of the generated series, which means that the output stream of FRNG should be balanced. So the probability of appearing of the output bits of each used LFSR in the output series should be approximately equal to each other [7]. In our case, with using four LFSRs in constructing the FRNG, the probability of appearing bits of each one of them in the out stream should be as close as possible to (1/4): $P(out_{sys}=out_{LFSR1}) \cong P(out_{sys}=out_{LFSR2}) \cong P(out_{sys}=out_{LFSR3}) \cong P(out_{sys}=out_{LFSR4}) \cong 0.25$, regarding the acceptance value of difference $\varepsilon \geq |P - 0.25|$ that should be previously defined.

Balancing the output sequences can be achieved by tuning the membership functions (MFs) that used to represent the terms of the fuzzy linguistic variables f_0 and $|f_1 - f_2|$ for each LFSR separately. As we found in [4] that the rate of appearing of output bits of each used LFSR mainly depends on the degree of characteristic primitive polynomial and the configurations of MFs that are associated with the related linguistic variables f_0 and $|f_1 - f_2|$.

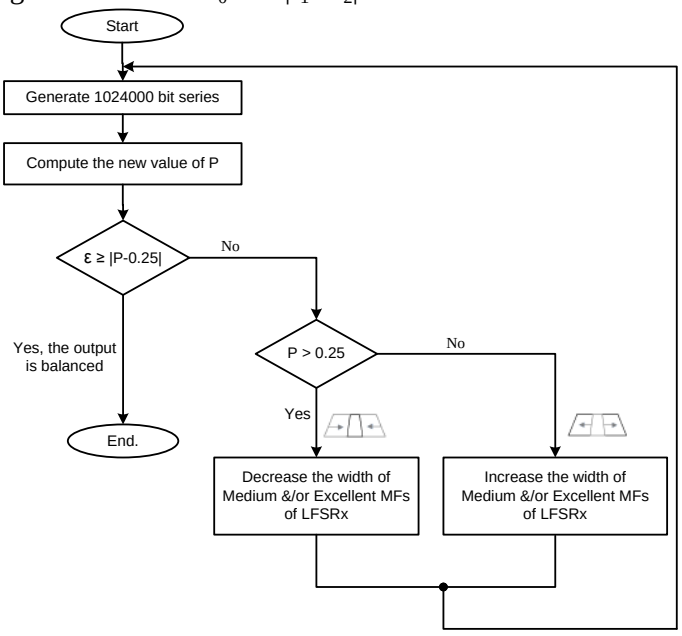

Figure 2. The algorithm of tuning MFs of linguistic variables related to LFSRx considering balance of generated sequence.

So in practical we should make some changes in the configurations of MFs of every LFSR in order to get the desired balanced version of FRNG. As described in [4] every linguistic variable has three membership functions (Low, Medium, High) for the first variable f_0 and (Excellent, Good, Bad) for the second $|f_1 - f_2|$, but there are two MFs that have more sufficient influence on probability of appearing of related LFSR' bits, they are the Medium MF of first variable f_0 and the Excellent MF of the second linguistic variable $|f_1 - f_2|$. It's worth mentioning that the other MFs of both linguistic variables affect the calculated value of probability but their effect is smaller and it's very useful when resulting value of probability P is not too far from 0.25. As illustrated in 'figure 1', if we want to increase the rate of appearing LFSRx' bits (x=1,..,4) in the output stream we should increase the width

of (Medium) MF of the first variable f_0 and/or increase the width of (Excellent) MF of the second variable $|f_1-f_2|$ that are associated with LFSRx. Then we evaluate the balance of the output sequence by generating a new sequence with a size of 1024000 bits and compute the new value of $P(out_{sys}=out_{LFSRx})$, so according to the obtained value the process of tuning will continue repeatedly or finish as reaching the accepted value of probability that fits the condition of stopping $\varepsilon \geq |P - 0.25|$ that defined by 0.0050. The process of tuning should be applied on every LFSR separately until reaching the relatively accepted results for all of them. As a practical example, we selected four primitive polynomials that have previously defined type by formula (1) with the degrees (89,97,113,127), then new FRNG constructed and initiated then we started the process of tuning the MFs of linguistic variables f_0 and $|f_1-f_2|$ for each LFSR separately to get the balanced version of FRNG. 'Table 1' contains the obtained results with ε=0.0050. All the numerical experiments made using MATLAB environment (version 7.14.0.739 (R2012a)).

Table 1. The resulting settings of MFs of balancing FRNG with 4LFSRs.

Characteristic polynomials of LFSRx used in constructing the FRNG		*The settings of MFs after tuning*		*resulting $P(out_{sys}=out_{LFSRx})$ after tuning*
LFSRx	$(b_1, b_2, ..., b_m, n)$	f_0	$\|f_1 - f_2\|$	
LFSR1	(1,5,10,17,39,89)	Low:{0,..,7} Medium:{8,..,24} High={25,..,32}	Excellent:{0,1,2} Good:{3,4,5} Bad:{6,..,10}	0. 2505
LFSR2	(1,4,7,20,53,97)	Low:{0,..,11} Medium:{12,..,18} High:{19,..,32}	Excellent:{0,1} Good:{2,3} Bad:{4,..,10}	0.2518
LFSR3	(2,3,10,23,48,113)	Low:{0,..,14} Medium:{15,16,17} High:{18,..,32}	Excellent:{0} Good:{2} Bad:{3,..,10}	0.2452
LFSR4	(1,2,4,13,102,127)	Low:{0,..,11} Medium:{12,..,21} High:{22,..,32}	Excellent:{0} Good:{1,2,3} Bad:{4,..,10}	0.2525

The resulting version of FRNG has a very long period

$T_s=T_1 \cdot T_2 \cdot T_3 \cdot T_4=(2^{89}-1)(2^{97}-1)(2^{113}-1)(2^{127}-1) \cong 2^{426}$

Finally, it's very important to mention that the resulting configurations of the MFs related to each of used LFSR definitely depends on the initiate state of LFSRs (the seeds).

4. Constructing the FRNG using eight LFSRs

Another more interesting example is a new version of FRNG that constructed using eight LFSRs. It's practically considered more complicated than the previous example that contains four LFSRs, because of the complexity of the process of tuning the MFs and finding the suitable configurations in order to get the balanced FRNG, considering the new value of probability $P(out_{sys}=out_{LFSRx}) \cong 1/8$. So the resulting generator will be very secure and very close to true random number generator.

Firstly we selected eight characteristic primitive polynomials that satisfy equation (1) and the associated conditions, with the degrees (61,71,79,89,97,103,113,127). Then a new version of FRNG constructed and initiated then we started the process of tuning the MFs of linguistic variables using the same method that used with four LFSRs example, considering the new value of probability $P(out_{sys}=out_{LFSRx}) \cong 0.125$ to get the balanced version of FRNG.

It's worth mentioning that all used polynomials accurately selected and tested to verify their primitivity using *Mathematica* software where we programmed algorithm for finding previously described polynomials and passing them through primitivity tests to pick up only primitive ones.

The obtained results of tuning process of FRNG with 8LFSRs concerning the acceptance value ε=0.0025 are shown in table 2 below.

Table 2. The resulting settings of MFs of balancing FRNG with 8LFSRs.

Characteristic polynomials of LFSRx used in constructing the FRNG		*The settings of MFs after tuning*		*resulting $P(out_{sys}=out_{LFSRx})$ after tuning*
LFSRx	$(b_1, b_2, ..., b_m, n)$	f_0	$\|f_1 - f_2\|$	
LFSR1	(2,3,15,39,61)	Low:{0,..,11} Medium:{12,..,21} High:{22,..,32}	Excellent:{0,1} Good:{2,3} Bad:{4,..,10}	0.1266
LFSR2	(2,4,9,17,37,71)	Low:{0,..,16} Medium:{17,..,28} High:{29,..,32}	Excellent:{0,1,2} Good:{3,4} Bad:{5,..,10}	0.1230
LFSR3	(2,5,8,19,41,79)	Low:{0,..,16} Medium:{17,18,19} High:{20,..,32}	Excellent:{0,1,2} Good:{3,4} Bad:{5,..,10}	0.1244
LFSR4	(1,5,10,17,39,89)	Low:{0,..,10} Medium:{11,..,18} High:{19,..,32}	Excellent:{0} Good:{1,2,3} Bad:{4,..,10}	0.1265
LFSR5	(1,4,7,20,53,97)	Low:{0,..,13} Medium:{14,..,19} High:{20,..,32}	Excellent:{0} Good:{1} Bad:{2,..,10}	0.1258
LFSR6	(1,2,9,19,70,103)	Low:{0,..,18} Medium:{19,...,22} High:{23,..,32}	Excellent:{0,1,2} Good:{3,4,5} Bad:{6,..,10}	0.1228
LFSR7	(2,3,10,23,48,113)	Low={0,..,15} Medium={16,...,19} High={20,..,32}	Excellent:{0} Good:{1,2,3} Bad:{4,..,10}	0.1275
LFSR8	(1,2,4,13,102,127)	Low:{0,..,18} Medium:{19,...,22} High:{23,..,32}	Excellent:{0,1} Good:{2,3} Bad:{4,..,10}	0.1234

The resulting version of FRNG has a very long period T_s:

$$T_s = T_1 \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6 \cdot T_7 \cdot T_8$$
$$= (2^{61}-1)(2^{71}-1)(2^{79}-1)(2^{89}-1)(2^{97}-1)(2^{103}-1)(2^{113}-1)(2^{127}-1)$$
$$\cong 2^{740}$$

With such huge period the FRNG became very close to real RNG and secure against most known attacks, and it can be used safely and for a long time in the field of data protection especially with the big progress of data base storing techniques and appearing the big data bases projects with the necessity of securing it.

5. Conclusion

In this paper we have introduced a new versions of the proposed FRNG that emphasized its security and made the generated sequences have very high immunity against correlation attacks, with a very high linear complexity due two using larger number of LFSRs that makes them more secure against

most known algebraic attacks. Getting benefit of using such type of primitive polynomials that defined by equation (1) makes the proposed generator has less power consumption, and passing all the randomness tests included in NIST and DIEHARD packets, that proved very high statistical properties of the generated sequences by the new versions of FRNG. All these properties make the proposed FRNG has very high quality and make it very attractive and fits all the requirements of most applications like modeling and simulation, gaming industry, cryptography, key generation, authentication protocols etc.

Finally we can conclude that after improving the quality of FRNG by increasing the period of the generated sequence the proposed pseudo-random numbers generator based on fuzzy logic (FRNG) becomes very close to true random generator.

6. References

[1] L'Ecuyer, P. Random Number Generators: Selection Criteria And Testing in Lecture Notes in Statistics / P. L'Ecuyer, P. Hellekalek // Lecture Notes in Statistics. – New York: Springer-Verlag, 1998. – Vol. 138 – P. 223-266.

[2] Zadeh, L.A. Fuzzy Sets, Fuzzy Logic, Fuzzy Systems / L.A. Zadeh, et al. // World Scientific Press – 1996.

[3] Anikin, I.V. Fuzzy stream cipher system / I.V. Anikin, K. Alnajjar // Proc. Int. Siberian Conf. on Control and Communications. – Omsk, 2015. – P. 64-68.

[4] Anikin, I.V. Pseudo-random number generator based on fuzzy logic / I.V. Anikin, K. Alnajjar // Proc. Int. Siberian Conf. on Control and Communications. – Moscow, 2016. – P. 68-72.

[5] Knuth, D.E. The art of computer programming / D.E. Knuth // Boston: Addison-Wesley Longman Publishing Co., 1997. – Vol. 2.

[6] Wang, L.-T. On Designing Transformed Linear Feedback Shift Registers with Minimum Hardaware Cost / L.-T. Wang, N.A. Touba, R.P. Brent, H. Xu, H. Wang // University of Texas at Austin, Technical report, – 2011. – UT-CERC-12-03.

[7] Siegenthaler, T. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications / T. Siegenthaler // IEEE Transactions on Information Theory. –1984. – Vol. 30(5). – P. 776-780.