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

I V Anikin¹ and K Alnajjar¹

¹Kazan National Research Technical University-KAI, K. Marks str. 68, Kazan, Russia, 420111
e-mail: anikinigor777@mail.ru, khkazan@mail.ru

Abstract. We consider previously suggested by authors pseudo-random number generator based on fuzzy logic (FRNG) to increase its quality by increasing the period of generated series. To increase the period of FRNG we include 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 considering the randomness and security of the generated sequences. The resulted new versions of FRNG generate pseudo-random numbers which are more close to true random series that make it more suitable and attractive for many applications such as data security and cryptography.

1. Introduction

Nowadays, it is well known that pseudorandom number generations have 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, high immunity against correlation attacks, and its 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 linear feedback shift registers (LFSRs) using a non-linear function, are the most common long-period generators available at the present time [1].

![Figure 1. General structure of proposed FRNG.](image)

In [2,3] authors proposed pseudorandom number generator based on fuzzy logic techniques [2] (FRNG), where a new non-linear function created to combine number of LFSRs as depicted in ‘figure...
1’. Firstly the outputs of LFSRs pass through number of buffers that have same size (32bits), where an estimation of two fuzzy statistical variables over them 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 consecutive ones $f_1$ and the number of runs consisting of two zeros $f_2$). These variables were accurately selected depending on Golomb's postulates [4]. Then a group of fuzzy If-Then rules evaluates the statistical state in every buffer at the moment and selects the LFSR with the best statistical properties and passes its LSB to the output of the system at the moment. Then a new iteration to produce a new bit begins after shifting the LFSRs one bit to the right and updating the contents of buffers respectively and evaluation the statistical situation of them and so on.

In [3,5] we investigated the parameters of FRNG and proved that the suggested generator produces a pseudorandom series with good statistical properties by testing them using the most powerful randomness tests packets (NIST, DIEHARD) [5]. In our previous researches we used only two LFSRs in constructing the suggested 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 show through a simple example that the suggested FRNG generates a sequences with a maximum period that is defined by multiplying the periods of LFSRs used in constructing it. Then we will study the case of combining four LFSRs then the case of using eight LFSRs. The new versions of FRNG have larger periods and produce pseudorandom sequences with high level of randomness that makes it more suitable and attractive for many applications related to data security and cryptography.

### 2. Studying the period a very simple version of FRNG

It’s necessary at first to explain the principle of work of FRNG using very short two LFSRs with a following simple characteristic primitive polynomials with small degrees (2,5):

$$P_1(x) = x^2 + x + 1; \quad T_1 = 2^2 - 1 = 3$$
$$P_2(x) = x^3 + x + 1; \quad T_2 = 2^3 - 1 = 7$$

Then fined the period of the output sequence and how it’s related to the periods of used polynomials ($T_1$, $T_2$). After initiating the LFSRs with ([1 0], [0 1 0]) respectively, we will get the following periodic sequences:

For LFSR1: 10110110110110110110111011011011011011011011011...
For LFSR2: 0101110010111001011100101110...

Then after filling the related buffers (here they are taken with 8 bits size for simplicity) the suggested system evaluates the statistical situation of every buffer applying the functions $f_0$ and $|f_1-f_2|$, then the resulted sequences fuzzificated and then a group of fuzzy If-Then rules applied to the output fuzzy sequences as showed in table 1.

| $f_0$  | $|f_1-f_2|$ | fuzzy out |
|-------|------------|-----------|
| Low   | Excellent  | Bad       |
| Low   | Good       | Good      |
| Low   | Bad        | Bad       |
| Medium| Excellent  | Best      |
| Medium| Good       | Good      |
| Medium| Bad        | Good      |
| High  | Excellent  | Bad       |
| High  | Good       | Good      |
| High  | Bad        | Bad       |

Then the system compares the results and decide which LSB of buffers will be selected as the output of the generator at this moment and so on. If the results of estimations is equal for the two
buffers the system will pass the LSB of the buffer that is associated with the shorter LFSR (in our case the LFSR1). Applying previously described process repeatedly we will get the following sequence of bits:

The output sequence of FRNG: 011011001011000111010011011001011000111010…

As seen from the resulted sequence that it repeats itself ever 21 bits. So the output stream is defined by

\[ T_s = T_1 \cdot T_2 \]  

Also 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 and the degrees are mutual primitive numbers. So the period of the output stream \( T_s \) of the general structure of FRNG depicted in ‘figure 1’ is defined by the following formula:

\[ T_s = T_1 \cdot T_2 \cdots \cdot T_n = (2^{d_1} - 1)(2^{d_2} - 1)L(2^{d_n} - 1) \]  

where \( d_1, d_2, \ldots, d_n \) represent the degrees of characteristic primitive polynomials of used LFSRs respectively.

3. 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 [6]. 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 [7], it guides us to a good 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:

\[
P(x) = (1 + x^{b_1})(1 + x^{b_2})(1 + x^{b_3}) + x^{m}
\]

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 \((b_1, b_2, \ldots, b_m, n)\) should satisfy the following conditions [7]:

\[
\begin{align*}
&b_1 \geq 1, b_1 < b_2, (b_1 + b_2) < b_3, \ldots, b_m, \\
&(b_1 + b_2 + \ldots + b_m) < b_m, \\
&(b_1 + b_2 + \ldots + b_m) < n
\end{align*}
\]

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 such type of polynomials (4) defined by \( t=2^m+1 \), so with \( m=5 \) we will get polynomials having high diffusion capacity due to the Hamming weight \( t=33 \) 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 \((b_1, b_2, \ldots, b_m, n)\). For example, the tuple \((1, 4, 7, 20, 53, 97)\) refers to the following primitive polynomial:

\[
P(x) = (1 + x^1)(1 + x^4)(1 + x^7)(1 + x^{20})(1 + x^{53}) + x^{97}
\]

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 increase efficiency and the security of the designed generator.

4. Constructing the FRNG using four LFSRs

Firstly we will construct our new FRNG with 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 [6]. In our case, with using four LFSRs in constructing the FRNG,
the probability of appearing bits of each one of them in the output stream should be as close as possible to 0.5: \( P(out_{sys} = out_{LFSRx}) \leq 0.5 \) where \( x \) differ from 1 to 4.

Balancing the output series can be achieved via tuning the membership functions of the fuzzy linguistic variables \( f_0 \) and \( |f_1 - f_2| \), for each LFSR separately. As we found in [5] the probability of appearing of the output bits of each used LFSR mainly depends on the degree of characteristic primitive polynomial and the configurations of membership functions MFs that are associated with the related linguistic variables \( f_0 \) and \( |f_1 - f_2| \). So in practice 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 [5] 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 \( |f_1 - f_2| \). Other MFs of both linguistic variables affect the calculated value of probability but their effect is smaller and it's very useful when the resulting value of probability \( P \) is not too far from 0.5 (fine tuning). So if we want to increase the probability of appearing LFSRx' bits (\( x = 1, \ldots, 4 \)) 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 LFSRs. Then we evaluate the balance of the output series by generating a new series with a size of 1024000 bits and compute the new value of probability \( P(out_{sys} = out_{LFSRx}) \), after that according to the obtained value the process will continue repeatedly until reaching \( P \) within the margin of the predefined acceptance value \( \varepsilon \geq |P - 0.5| \) that defined by 0.05. ‘Figure 2’ shows algorithm of tuning the MFs.

![Figure 2](image)

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

The process of tuning should be applied on every LFSR separately until reaching the relatively accepted results for all of them. As an example, we selected four primitive polynomials that have previously defined type by formula (4) and satisfy the related conditions (5) with the degrees (89,97,113,127) these polynomials are represented in (7), then the new FRNG constructed and initiated then we started the process of tuning of the MFs of linguistic variables \( f_0 \) and \( |f_1 - f_2| \) for each LFSR separately according ‘figure 2’ to get the balanced version of FRNG.
The IV International Conference on Information Technology and Nanotechnology

IOP Conf. Series: Journal of Physics: Conf. Series 1096 (2019) 012193

IOP Publishing
doi:10.1088/1742-6596/1096/1/012193

Table 2 contains the obtained results with \( \varepsilon = 0.05 \). All the numerical experiments have been made using MATLAB environment (version 7.14.0.739 (R2012a)) [8].

The resulting version of FRNG has a long period:

\[
T_f = T_1 T_2 T_3 T_4
\]

\[
= (2^{89} - 1)(2^{57} - 1)(2^{113} - 1)(2^{127} - 1)
\]

\[
\approx 2^{426} \approx 10^{128}
\]

Finally, we have to mention that the resulting configurations of the membership functions of used linguistic variables \( \{f_0, |f_1 - f_2|\} \) of every LFSR depend on the initiate state of used LFSRs (the seeds).

### Table 2. The resulting balanced FRNG with four LFSRs after tuning the MFs.

| Primitive polynomials used in constructing the FRNG | Configurations of MFs of \( \{f_0\} \) | Configurations of MFs of \( \{f_1,f_2\} \) | \( P(\text{out}_{sys}=\text{out}_{LFSRs}) \) |
|-----------------------------------------------|---------------------------------|---------------------------------|--------------------------------|
| LFSR1 \( (1,5,10,17,39,89) \) | Low: \{0,...,7\} | Excellent: \{0,1,2\} | 0.51 |
| Medium: \{8,...,24\} | Good: \{3,4,5\} | | |
| High: \{25,...,32\} | Bad: \{6,...,10\} | | |
| LFSR2 \( (1,4,7,20,53,97) \) | Low: \{0,...,11\} | Excellent: \{0\} | 0.48 |
| Medium: \{12,...,18\} | Good: \{2\} | | |
| High: \{19,...,32\} | Bad: \{4,...,10\} | | |
| LFSR3 \( (2,3,10,23,48,113) \) | Low: \{0,...,14\} | Excellent: \{0\} | 0.54 |
| Medium: \{15,16,17\} | Good: \{2\} | | |
| High: \{18,...,32\} | Bad: \{3,...,10\} | | |
| LFSR4 \( (1,2,4,13,102,127) \) | Low: \{0,...,11\} | Excellent: \{0\} | 0.49 |
| Medium: \{12,...,21\} | Good: \{1,2,3\} | | |
| High: \{22,...,32\} | Bad: \{4,...,10\} | | |

5. Constructing the FRNG using eight LFSRs

Another interesting example is a version of FRNG that constructed with eight LFSRs. It’s more complicated than the previous one because of the time complexity of the process of tuning the MFs till getting the balanced FRNG. But the resulting generator will be very secure and very close to the true random number generator due to its huge period.

Firstly we selected eight characteristic primitive polynomials of the defined type (4) and that satisfy the related conditions (5) with degrees \( (61,71,79,89,97,103,113,127) \), these polynomials are described by (8):

\[
P_1(x) = (1 + x^2)(1 + x^3)(1 + x^{15})(1 + x^{39}) + x^{41}
\]

\[
P_2(x) = (1 + x^2)(1 + x^3)(1 + x^4)(1 + x^{14})(1 + x^{37}) + x^{71}
\]

\[
P_3(x) = (1 + x^2)(1 + x^3)(1 + x^5)(1 + x^{10})(1 + x^{41}) + x^{79}
\]

\[
P_4(x) = (1 + x)(1 + x^3)(1 + x^4)(1 + x^{15})(1 + x^{39}) + x^{89}
\]

\[
P_5(x) = (1 + x)(1 + x^3)(1 + x^4)(1 + x^{10})(1 + x^{53}) + x^{97}
\]

\[
P_6(x) = (1 + x)(1 + x^3)(1 + x^4)(1 + x^{16})(1 + x^{30}) + x^{103}
\]

\[
P_7(x) = (1 + x^2)(1 + x^3)(1 + x^{10})(1 + x^{23})(1 + x^{48}) + x^{113}
\]

\[
P_k(x) = (1 + x)(1 + x^2)(1 + x^3)(1 + x^5)(1 + x^{102}) + x^{127}
\]

(8)
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 \( P(\text{out}_{\text{sys}} = \text{out}_{\text{LFSR}}) \approx 0.5 \) to get the balanced version of FRNG. Table 3 contains obtained results after finishing the tuning process with the acceptance value \( \epsilon = 0.050 \).

The resulting version of FRNG has a period:

\[
T_s = T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8
\]

\[
= (2^{61} - 1)(2^{71} - 1)(2^{79} - 1)(2^{89} - 1)(2^{97} - 1)(2^{103} - 1)(2^{113} - 1)(2^{127} - 1)
\]

\[
\approx 2^{740} \approx 10^{223}
\]

Table 3. The resulting balanced FRNG with eight LFSRs after tuning the MFs.

| Primitive polynomials used in constructing the FRNG | Configurations of MFs of \((f_0)\) | Configurations of MFs of \(|f_1 - f_2|\) | \(P(\text{out}_{\text{sys}} = \text{out}_{\text{LFSR}})\) |
|---------------------------------------------------|---------------------------------|---------------------------------|-----------------|
| LFSR1 \((2,3,15,39,61)\)                         | Low: \{0,..,11\}               | Excellent: \{0,1\}             | 0.54            |
|                                                 | Medium: \{12,..,21\}           | Good: \{2,3\}                  |                 |
|                                                 | High: \{22,..,32\}            | Bad: \{4,..,10\}              |                 |
| LFSR2 \((2,4,9,17,37,71)\)                       | Low: \{0,..,16\}              | Excellent: \{0,1,2\}           | 0.52            |
|                                                 | Medium: \{17,..,28\}          | Good: \{3,4\}                  |                 |
|                                                 | High: \{29,..,32\}            | Bad: \{5,..,10\}              |                 |
| LFSR3 \((2,5,8,19,41,79)\)                       | Low: \{0,..,16\}              | Excellent: \{0,1,2\}           | 0.48            |
|                                                 | Medium: \{17,18,19\}          | Good: \{3,4\}                  |                 |
|                                                 | High: \{20,..,32\}            | Bad: \{5,..,10\}              |                 |
| LFSR4 \((1,5,10,17,39,89)\)                      | Low: \{0,..,10\}              | Excellent: \{0\}               | 0.51            |
|                                                 | Medium: \{11,..,18\}          | Good: \{1,2,3\}                |                 |
|                                                 | High: \{19,..,32\}            | Bad: \{4,..,10\}              |                 |
| LFSR5 \((1,4,7,20,53,97)\)                       | Low: \{0,..,13\}              | Excellent: \{0\}               | 0.52            |
|                                                 | Medium: \{14,..,19\}          | Good: \{1\}                    |                 |
|                                                 | High: \{20,..,32\}            | Bad: \{2,..,10\}              |                 |
| LFSR6 \((1,2,9,19,70,103)\)                      | Low: \{0,..,18\}              | Excellent: \{0,1,2\}           | 0.48            |
|                                                 | Medium: \{19,..,22\}          | Good: \{3,4,5\}                |                 |
|                                                 | High: \{23,..,32\}            | Bad: \{6,..,10\}              |                 |
| LFSR7 \((2,3,10,23,48,113)\)                     | Low: \{0,..,15\}              | Excellent: \{0\}               | 0.54            |
|                                                 | Medium: \{16,..,19\}          | Good: \{1,2,3\}                |                 |
|                                                 | High: \{20,..,32\}            | Bad: \{4,..,10\}              |                 |
| LFSR8 \((1,2,4,13,102,127)\)                     | Low: \{0,..,18\}              | Excellent: \{0,1\}             | 0.52            |
|                                                 | Medium: \{19,..,22\}          | Good: \{2,3\}                  |                 |
|                                                 | High: \{23,..,32\}            | Bad: \{4,..,10\}              |                 |

With such period the proposed FRNG became very close to real RNG and more secure against most known attacks. 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 increasing of the necessity of securing it.

6. Conclusion
In this paper we have introduced a new versions of the FRNG that emphasized its security and made the generated sequences have very high immunity against correlation attacks, with a very high linear complexity due to 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 formula (4) 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 FRNG.

All these properties make the proposed FRNG has very high quality and make it 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 sequences the proposed pseudo-random numbers generator based on fuzzy logic (FRNG) becomes very close to true random generator.

7. References
[1] L’Ecuyer P and Hellekalek P 1998 Random Number Generators: Selection Criteria And Testing Lecture Notes in Statistics 138 223-266
[2] Anikin I V and Alnajjar K 2015 Fuzzy stream cipher system Proc. Int. Siberian Conf. on Control and Communications (Omsk)
[3] Anikin I V and Alnajjar K 2016 Pseudo-random number generator based on fuzzy logic Proc. Int. Siberian Conf. on Control and Communications (Moscow)
[4] Henk C A and van Tilborg 2000 Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial 528 492
[5] Anikin I V and Alnajjar K 2018 Primitive polynomials selection method for pseudo-random number generator Journal of Physics: Conf. Series 944 012003
[6] Siegenthaler T 1984 Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications IEEE Transactions on Information Theory. 30(5) 776-780
[7] Wang L T Nur A T Richard P Brent Xu H and Wang H 2011 On Designing Transformed Linear Feedback Shift Registers with Minimum Hardware Cost Technical report: UT-CERC-12-03
[8] MATLAB Documentation MathWorks Retrieved 14 August 2013