Secure gamma generation for stream cipher based on fuzzy logic

K. Alnajjar[1], I.V. Anikin[2]

[1]Higher Institute for Applied Science and Technology, Damascus, Syria
[2]Information Security Systems Department, Kazan National Research Technical University named after A.N. Tupolev-KAI, K. Marx str. 14, Kazan, Russia, 420111

Abstract. In this paper, we propose a new approach for secure bit stream gamma generation based on the concept of fuzzy logic and linear feedback shift registers (LFSRs), combined in pseudorandom number generator based on fuzzy logic (FPRNG). The original idea for ensuring security for gamma generation is based on the initial states (the seeds) of LFSRs used in the generator. In this paper, we suggested the updated version of FPRNG. We also improve gamma security by accompaniment seed some valuable secure information. We found that configurations of membership functions (MFs) in FPRNG can significantly affect the randomness of the output stream of the developed generator. We can increase the gamma security as well as stream cipher security by keeping these configurations in secret additionally to seeds.

1. Introduction

Generating of pseudo-random streams with high statistical properties is still a very challenging job for stream ciphers. Such a pseudo-random stream could be used as a bit stream or gamma in such ciphers and their statistical properties mostly define the security level of the stream cipher system. Most of the good pseudo-random number generators use non-linear functions to combine the output of several LFSRs [7]. In the work [1] we developed a pseudo-random number generator based on fuzzy logic [8] [9] (FPRNG) which gives us a lot of benefits to achieving non-linearity. 'Figure 1' depicts the general structure of FPRNG. As shown in 'figure 1', the generator depends on two linguistic variables (f_0: number of ones in the buffer, $|f_1-f_2|$: the difference between the number of blocks "0110" and the number of gaps "1001" in the buffer) to estimate the statistical status of the data included in every buffer. Then a group of fuzzy if-then rules helps to decide which of these buffers is the best at the considered moment and passes its last bit to the output of the generator. A new step starts when FPRNG updates the contents of buffers by running the LFSRs one step.

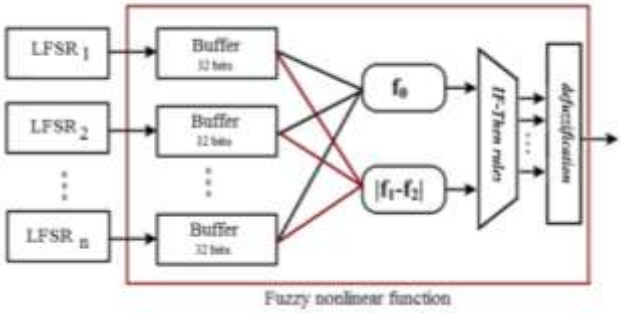

Figure 1. The structure of developed FPRNG.

In [2], we studied the immunity of FPRNG against correlation attacks and explained the iterative tuning process of linguistic variables. The tuning process guarantees the security of the generated stream against correlation attacks because it takes into account the balance of the output stream. We also proved that the suggested numerical method for selecting primitive characteristic polynomials for LFSRs practically improves the statistical properties of the generated bit stream and increases the performance of FPRNG. The chosen polynomials have the general form defined by equation (1) and satisfy the specified conditions that we explained in detail in [2].

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

This kind of polynomials has a relatively high Hamming weight (high number of non-zero coefficients). This property in its role increases the diffusion capacity, and at the same time, it reduces the cost of hardware implementation as it requires less number of XOR gates. Previous versions of FPRNG have the same size for all buffers of used LFSRs. In [1], we studied FPRNG's parameters to pick up the best values for them. We found that the best value of the size of the buffer regarding the statistical properties of the generated stream is equal to 32 bits. In the next section, we discuss this parameter more precisely, considering the degree of the attached characteristic polynomial of LFSR.
We organized this paper as follows. Section 2 studies the relationship between the degree of the selected primitive polynomial and the size of the attached buffer. Section 3 describes the role that MFs configurations play in the developed FPRNG and how they affect the generated output bit stream. In Section 4, we discuss the motivated properties of MFs configurations that made us consider them similar to a secure state. Finally, the conclusion summarizes the results of our research.

2. Studying the relationship between the degree of the selected primitive polynomial and the size of the attached buffer

We try to look for a suitable size for each buffer by investigating the relationship between the degree of the used primitive polynomial and the length of the linked buffer measured in bits. In [2], we have built a table containing the ranges of possible degrees regarding the selected value of m (the number of multiplicative terms in the polynomial expression) and t (the number of non-zero coefficients). As a result of this section, we add a new column to the built table containing the recommended sizes of the buffer.

We used three statistical rules to get the proper length of the buffer. We got them from the first and the second randomness postulates of Golomb [5]:

1. The number of zeros and the number of ones is as equal as possible in the buffer.
2. Half of the runs in the buffer of length 2 are gaps, the other blocks.
3. Half of the runs in the buffer of length 3 are gaps, the other blocks.

Practically, we selected one primitive polynomial from every range of the degrees as a characteristic polynomial for LFSR. Then we compare between the various values of the buffer: {24,32,40,48,56,64}. Table 1 contains the selected polynomials briefly described using the parameters (b_1, b_2, …, b_m, n) of (1).

Table 1. Selected primitive polynomials.

M	*Range of degrees*	*Selected polynomial* (b_1,b_2,…,b_m,n)
3	19-34	(1,6,15,23)
4	35-66	(1,6,11,22,43)
5	67-130	(1,5,9,21,41,89)
6	131-258	(1,5,7,15,29,129,191)
7	259-514	(1,2,4,8,16,33,193,419)
8	515-1026	(1,2,5,9,18,36,72,298,929)
9	1027-2050	(1,2,4,8,16,32,64,129,807,1163)

To remove confusion, we write one these polynomials in its expanded form (2):

$$P(x) = (1 + x)(1 + x^5)(1 + x^7)(1 + x^{15})(1 + x^{29})(1 + x^{129}) + x^{191}$$

$$
\begin{aligned}
&= 1 + x + x^5 + x^6 + x^7 + x^8 + x^{12} + x^{13} + x^{15} + x^{16} + x^{20} + x^{21} + x^{22} \\
&+ x^{23} + x^{27} + x^{28} + x^{29} + x^{30} + x^{34} + x^{35} + x^{36} + x^{37} + x^{41} + x^{42} \\
&+ x^{44} + x^{45} + x^{49} + x^{50} + x^{51} + x^{52} + x^{56} + x^{57} + x^{129} + x^{130} \\
&+ x^{134} + x^{135} + x^{136} + x^{137} + x^{141} + x^{142} + x^{144} + x^{145} + x^{149} \\
&+ x^{150} + x^{151} + x^{152} + x^{156} + x^{157} + x^{158} + x^{159} + x^{163} + x^{164} \\
&+ x^{165} + x^{166} + x^{170} + x^{171} + x^{173} + x^{174} + x^{178} + x^{179} + x^{180} \\
&+ x^{181} + x^{185} + x^{186} + x^{191}
\end{aligned} \tag{2}
$$

So, we performed 9×6 numerical experiments using Wolfram Mathematica 10.0 software, to find out the best value of buffer size. In every experiment, after initiating the selected LFSR, we generate 1 Mbits. Then we evaluate the statistical prosperities of the resulting bitstream using previously mentioned three statistical rules. Thus, we built five functions to assess the statistical situation of the buffered data. Table 2 briefly describes these functions.

Table 2. Description of functions used to evaluate statistical status of the buffers.

function	*Description*
f_0	Calculates the number of ones in the buffer.
f_{00}	Calculates the number of runs consisting of two consecutive zeros in the buffer.
f_{11}	Calculates the number of runs consisting of two consecutive ones in the buffer.
f_{000}	Calculates the number of runs consisting of three consecutive zeros in the buffer.
f_{111}	Calculates the number of runs consisting of three consecutive ones in the buffer.

It is worth mentioning, that all functions of table 2 should practically follow the normal distribution (Gaussian). ‘Figure 2’ illustrates a practical example, where we generated 1 Mbits using LFSR with the characteristic polynomial (1,5,7,15,29,129,191), and the chosen size of buffer equals 64 bits. Then we built the corresponding histogram to the third function of table 2 (f_{11}).

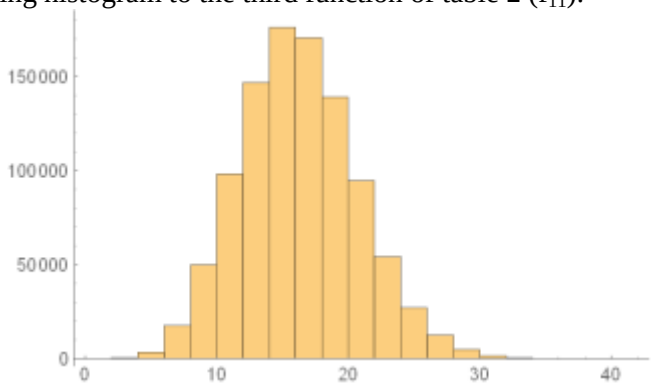

Figure 2. Statistical distribution of f_{11} values with 64-sized buffer.

As shown in ‘figure 2’, the values of f_{11} differ from 1 to 32. And the most frequent value is (12). For each polynomial, we performed six numerical experiments to cover all possible values of buffer length. As a result, we determined the acceptable values of buffer’s size that satisfy all mentioned statistical rules. Table 3 summarizes the obtained results. It is clear from Table 3, that there is more than one value satisfying the predefined statistical rules. The minimum possible value of the buffer's size has an inverse proportional relationship with the degree of selected LFSR. Hence, as the degree of the polynomial increases, the minimum size of the buffer decreases. This means that we have more possible choices for this variable as we develop a new version of FPRNG. It is worth mentioning that the obtained results are very worthy from the cryptanalysis point of view because it makes the developed generator more secure.

Table 3. Recommended sizes of the buffer.

m	*Range of degrees*	*Selected polynomial* $(b_1,b_2,\ldots,b_m,n)$	*Recommended buffer sizes*
3	19-34	(1,6,15,23)	56-64
4	35-66	(1,6,11,22,43)	48-64
5	67-130	(1,5,9,21,41,89)	40-64
6	131-258	(1,5,7,15,29,129,191)	40-64
7	259-514	(1,2,4,8,16,33,193,419)	32-64
8	515-1026	(1,2,5,9,18,36,72,298,929)	32-64
9	1027-2050	(1,2,4,8,16,32,64,129,807,1163)	24-64

3. The membership functions and their role in the developed FPRNG

In [3], we explained the MFs tuning process regarding the correlation between output bitstream of every used LFSR and the output of FPRNG. The success of the correlation attack depends on the variances of probabilities of appearing bits of every used $LFSR_x$ in the output sequence of the generator $P(out_{FPRNG} = LFSR_x)$, where *x* refers to the LFSR index. The value of probability for any used $LFSR_x$ should be as close as possible to 0.5. In [1], we found that best number of MFs for every linguistic variable (f_0, $|f_1-f_2|$) is equal to 3. There are three MFs for the first linguistic variable f_0: (Low, Medium, High), and three for the second $|f_1-f_2|$: (Excellent, Good, Bad). Then in [3,4], we discussed the results of tuning of the configurations of MFs regarding the immunity of the generated bit stream facing the correlation attacks. Configurations of MFs means defining number of elements belonging to the fuzzy group assigned with each MF, and also managing intersections (overlapping sections) between them. 'Figure 3' represents an example of the described MFs.

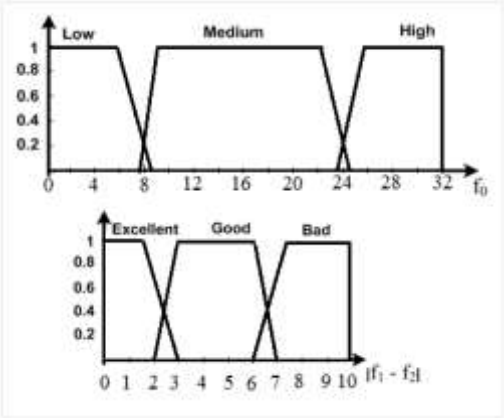

Figure 3. Configurations of MFs for (f_0, $|f_1-f_2|$).

Also, in [3], we investigated the relationship between the degree of used polynomial and the settings of related three MFs of every linguistic variable. Practically, we predefine the value of (ε), that determines the tolerance value of the probability of appearing for every LFSR' bits in the output stream $|P(out_{FPRNG} = LFSR_x) - 0.5| \le \varepsilon$. The iterative process of tuning the MFs finishes when the generated output sequence fulfils this predefined condition for all used LFSRs. Generated bit stream should successfully pass all the randomness tests. By then, we have got one of the possible settings for the MFs. 'Figure 4' represents the block diagram of the MFs tuning process.

In the preceding section, we found that different buffer's lengths can be used with every LFSR. So, to build the new version of FPRNG we firstly select the characteristic primitive polynomials for every LFSR. Then we can use the table 3 to choose one of the recommended sizes of the buffers. After that, we initiate the used LFSRs and define the value of ε. Then we start running the tuning process described in 'figure 4'.

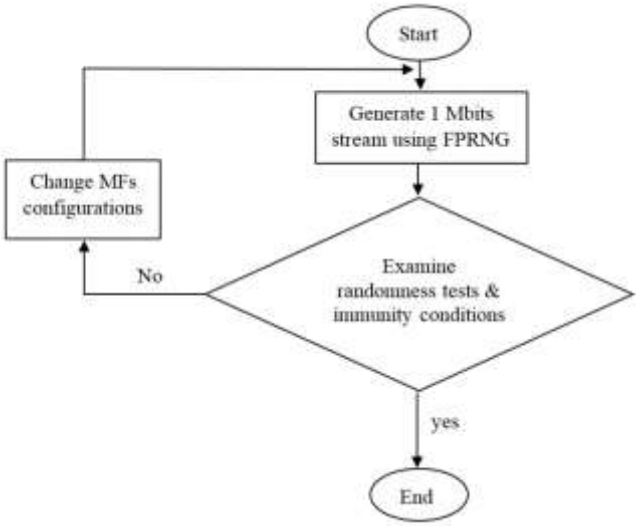

Figure 4. The block diagram of the MFs tuning process.

Practically, to reduce the testing time in every iteration, we used the most dominant five tests of the NIST STS full packet, which includes 15 tests. In earlier work, we determined these five tests according to three main criteria specified in [1]. It is worth to notice that the resulting configurations of the MFs depend on the beginning state of used LFSRs.

Table 4 contains the results of applying the explained process of tuning using the simplest scenario to construct FPRNG. The constructed generator consists of two LFSRs selected from two different ranges (with the intent to meet studying goals). We chose them utilizing table 3. The first characteristic polynomial is (1,6,11,22,43) with buffer size equals 64 bits and the second is (1,5,7,15,29,129,191) with buffer size equals 48 bits. The selected value for ε is 0.1.

Table 4. The resulting settings of new version of FPRNG after finishing of the tuning process.

Selected Primitive polynomials to construct the FPRNG		*buffer size*	*Resulting configurations of MFs related to (f_0)*	*Resulting configurations of MFs related to ($\lvert f_1-f_2 \rvert$)*
LFSR1	(1,6,11,22,43)	64	Low:{0,...,27} Medium:{28,...,45} High={46,...,64}	Excellent:{0,...,4} Good:{5,…,18} Bad:{19,..,26}
LFSR2	(1,5,7,15,29,129,191)	48	Low:{0,...,11} Medium:{12,...,39} High={40,...,48}	Excellent:{0,…,5} Good:{6,…,15} Bad:{16,..,21}

It's important to mention that we can find a lot of possible configurations of the MFs. The number of acceptable options increases in the developed FPRNG, as we can use different sizes for buffers. Table 5 contains other accepted settings of MFs.

Table 5. Other accepted settings for MFs.

Selected Primitive polynomials to construct the FPRNG		*buffer size*	*Resulting configurations of MFs related to (f_0)*	*Resulting configurations of MFs related to ($\lvert f_1-f_2 \rvert$)*
LFSR1	(1,6,11,22,43)	64	Low:{0,...,13} Medium:{14,...,49} High={50,...,64}	Excellent:{0,1,2} Good:{3,…,22} Bad:{23,..,26}
LFSR2	(1,5,7,15,29,129,191)	48	Low:{0,...,7} Medium:{8,...,42} High={43,...,48}	Excellent:{0,…,7} Good:{8,…,14} Bad:{15,..,21}

Table 6 represents the results of the tuning process using a more complicated structure of FPRNG. In this scenario, the generator consists of four LFSRs. We chose them depending on table 3. The first characteristic polynomial is (1,6,11,22,43) with buffer size equals 64 bits and the second is (1,5,10,17,39,89) with buffer size equals 56 bits. The third polynomial is (1,5,7,15,29,129,191) with size of buffer equals 48. The fourth is (1,2,4,8,16,33,193,419) and the selected buffer size is 40. The chosen value for ε is the same (0.1).

Table 6. One of the accepted options of the MFs configurations for the constructed FPRNG using 4 LFSRs.

Selected Primitive polynomials to construct the FPRNG		*buffer size*	*Resulting configurations of MFs related to (f_0)*	*Resulting configurations of MFs related to ($\lvert f_1-f_2 \rvert$)*
			Low:{0,...,9}	Excellent:{0,...,7}
LFSR1	(1,6,11,22,43)	64	Medium:{11,...,45}	Good:{8,…,20}
			High={46,...,64}	Bad:{21,..,27}
			Low:{0,...,18}	Excellent:{0,…,8}
LFSR2	(1,5,10,17,39,89)	56	Medium:{19,...,35}	Good:{9,…,19}
			High={36,...,56}	Bad:{20,..,25}
			Low:{0,...,13}	Excellent:{0,…,7}
LFSR3	(1,5,7,15,29,129,191)	48	Medium:{14,...,29}	Good:{8,…,14}
			High={30,...,48}	Bad:{15,..,21}
			Low:{0,...,10}	Excellent:{0,…,7}
LFSR4	(1,2,4,8,16,33,193,419)	40	Medium:{11,...,23}	Good:{8,…,14}
			High={24,...,48}	Bad:{15,..,21}

Finally, after finishing the tuning process, the constructed FPRNG should successfully pass all the randomness tests. We used the full packet of NIST STS [6] to estimate the randomness quality of developed FPRNG. Table 7 shows that the new version of FPRNG successfully passed all 15 statistical tests of randomness included in packet NIST STS. We used the same three statistical criteria used in [1-4] that helps in making the final decision about passing the estimated sequence the test or failing it. The first criterion intends comparing the mean and variance of calculated P-values with the mean (0.5) and variance (1/12) of the uniform distribution. The second criterion involves calculating the chi-square statistic with nine degrees of freedom. The third criterion intends computing the ratio of failed sequences to the total number of tested sequences.

Table 7. Results of testing FPRNG using NIST STS.

Test	Result	Test	Result
1. The Frequency (Monobit)	success	9. Maurer's "Universal Statistical"	success
2. Frequency Test within a Block	success	10. Linear Complexity	success
3. Runs Test,	success	11. Serial	success
4. Longest-Run-of-Ones in a Block	success	12. Approximate Entropy	success
5. Binary Matrix Rank	success	13. Cumulative Sums (Cusums)	success
6. Discrete Fourier Transform	success	14. Random Excursions	success
7. Non-overlapping Template Matching Test	success	15. Random Excursions Variants	success
8. Overlapping Template Matching	success		

4. Security of gamma generation

The final bit stream which is the output of updated FPRNG could be directly used as a gamma sequence in different stream ciphers. It has a very good randomness quality and could be considered as a good white noise which is the main requirement for gamma sequence. The original idea for ensuring security for gamma generation is based on the initial states (the seeds) of LFSRs used in the generator. In the case of FPRNG we can use the configuration of membership functions as additional secret.

During the process of MFs adjusting, we have noticed that a little change in the settings of MFs leads to significant impacts on the generated sequence of the constructed FPRNG. Mainly, when we apply these changes on the central MFs (Medium membership function for the first linguistic variable f_0, Excellent membership function for the second $|f_1-f_2|$). We have also noticed in the previous section that there are a lot of options to choose the configuration of MFs from. Using fuzzy logic theory gives us the possibility to have an adaptive structure of the proposed generator. It's impossible to restore any information about configurations of MFs from the generated gamma. We can say that non-linear function based on fuzzy logic has very high complexity. We can consider it as a one-way function.

All these features have motivated us to consider the configuration of MF as additional information to ensure the security of the gamma sequence. The security level of the developed generator will increase if we keep the configuration of MFs and the seeds of used LFSRs both in secret.

5. Conclusion

In this paper, we suggested the updated version of the pseudo-random number generator based on fuzzy logic. We discussed the relationship between the size of the buffer and the degree of the primitive characteristic polynomial of the attached LFSR. Basing on the obtained results, we have updated the structure of FPRNG by using different sizes of buffers. Then we have explained the tuning process of the MFs configurations considering two important properties: passing the randomness tests successfully, and the immunity of constructed generator against correlation attacks. The developed FPRNG is very close to the true random number generator. We can consider the obtained MFs configuration as secret information. In this case, we can increase the gamma security as well as stream cipher security by keeping these configurations in secret additionally to the seeds.

6. References

[1] Anikin, I.V. Pseudo-random number generator based on fuzzy logic / I.V. Anikin, K. Alnajjar // Proc. Int. Siberian Conf. on Control and Communications, 2016.

[2] Anikin, I.V. Primitive polynomials selection method for pseudo-random number generator / I.V. Anikin, K. Alnajjar // Journal of Physics: Conf. Series. – 2018. – Vol. 944. – P. 012003.

[3] Anikin, I.V. Increasing the quality of pseudo-random number generator based on fuzzy logic generator / I.V. Anikin, K. Alnajjar // Journal of Physics: Conf. Series. – 2019. – Vol. 1096. – P. 012193.

[4] Anikin, I.V. Studying the relationship between linguistic variables and the degrees of primitive polynomials used in pseudo-random number generator based on fuzzy logic / I.V. Anikin, K. Alnajjar // Journal of Physics: Conf. Series. – 2019. – Vol. 1096. – P. 012178.

[5] Henk, C.A. Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial / C.A. Henk, van Tilborg // Springer, 2000. – 492 p.

[6] NIST SP 800-22 Revision 1a // A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, 2010. – 131 p.

[7] Acevedo, O. LFSR characteristic polynomial and phase shifter computation for two-dimensional test set generation / O. Acevedo, D. Kagaris // Proceedings of 18th IEEE Latin-American Test Symposium, 2017.

[8] Zadeh, L.A. Fuzzy Sets // Information and Control. – 1965. – Vol. 8. – P. 338-353.

[9] Zadeh, L.A. Linguistic variables and approximate reasoning // Proc. Annual Symposium on Computer Applications in Medical Care. – 1982. – P. 787-791.