Efficient Random Number Generators (RNG) based on Nonlinear Feedback Shift Registers (NLFSR)

Lo'ai Tawalbeh Wafa Kanakri and Lina Ebbine
Computer Engineering Department at Jordan University of Science and Technology (JUST)Jordan.
email: tawalbeh@just.edu.jo, wkanakri@kaddb.com and lina_ebbini@yahoo.com

Abstract

Non-Linear Feedback Shift Registers (NLFSRs) have been proposed as an alternative to Linear Feedback Shift Registers (LFSRs) for generating pseudo-random sequences for stream ciphers. In this paper, we will study, investigate and implement a pseudo random-bit sequence generator based on NLFSRs. The implementation of the proposed scheme makes use of NLFSR to modify the feedback function in each iteration. In addition, the advantages of both LFSR and NLFSR are implemented in Grain code in this work. Several statistical tests are applied on the generated sequences to show that these sequences have good statistical distribution properties and high linear complexity compared to key streams generated by LFSR.

I. Introduction

Information security is of paramount importance to many institutions of our society: governments, military, financial, and businesses. Confidential information about research, products, financial status, customers, or employees, is nowadays processed and stored on computers, or transmitted to other computers over a non-secure networks. From here, there is an increasing demand to provide information security services (such as confidentiality, authentication and integrity) which is generated mainly by symmetric/public key encryption algorithms.

From the above, the random number generation is considered very important and have many applications in computer security. For example, the private keys for symmetric cryptosystems (both block and stream ciphers) should be generated randomly for security purposes [1]. For asymmetric cryptosystems, random number generators (RNGs) are also required to generate public/private key pairs.

There are two basic types of generators for producing random sequences [1, 2]: true random number generators (TRNGs) and pseudorandom number generators (PRNGs). TRNGs make use of a non-deterministic entropy source, along with some post-processing functions to produce randomness. PRNGs, however, use an input called seed, along with deterministic algorithms to generate multiple pseudorandom numbers. PRNGs are usually faster than TRNGs and so are preferable in applications requiring a large quantity of random numbers [1].

Several techniques for random number generation have been developed. These techniques include generating a random-bit sequence from the content of an image [2], and generating true random numbers based on mouse movement which is cheap, convenient and universal [3].

Two binary pseudorandom generators based on logistic chaotic maps for cryptographic applications were proposed in [4]. The proposed generators have strong cryptographic properties such as high sensitivity to initial values and parameters of the logistic maps together with the un-correlation, random-like and unpredictability.

Recent study [7] proposed a new synchronous stream cipher that based on fuzzy bit generator (FBG). The FBG cipher has been designed to produce key stream with guaranteed randomness properties; i.e. passing all the statistical tests of the National Institute of Standard and Technology (NIST). The FBG stream cipher is on the form of a non-linear feedback shift register.

Besides the RNGs, Feedback Shift Registers (FSRs) are also widely used in secret-key ciphering, mainly in stream ciphers. Their structure is hardware oriented which leads to fast cryptosystems implementations [5]. In fact, stream ciphers are the core mechanism of high speed network encryption, wireless communication, memory card encryption and television conference due to the characteristic of small error propagation and simplified standardization [6].

This work is intended to use Non-Linear Feedback Shift Register (NLFSR) to generate pseudo random numbers. A NLFSR is a common component in modern stream ciphers, especially in Radio-Frequency Identification (RFID) and smart card applications [8]. NLFSRs are known to be more resistant to cryptanalytic attacks than Linear Feedback shift Registers (LFSRs), although construction of large NLFSRs with guaranteed long periods remains an open problem. In the implementation level, our proposed scheme makes use of NLFSR to modify the feedback function in each iteration. Then, we combine the output of LFSR approach with NLFSR to generate key streams with preferable generating process.

The rest of this paper is organized as follows: Section 2 and 3 introduces overviews of the RNG and the Shift Registers, respectively. Section 4 presents the NIST approved tests for Randomness. The proposed RNGs are presented and discussed in detail is Section 5. The results, evaluation and comparisons are found in Section 6. Section 7 concludes this work.

II. Random Number Generation

In this section we present an overview of True Random Number Generation and Pseudo Random Number Generation.

A. *True Random Number Generation*

A truly random sequence consists of digits that have pattern anywhere (thus being incompressible), and are predicted by any expression [9]. There is also another definition of truly random sequences based on that the sequence must be generated by a process such as the radiation decay, thermal noise in electronics, or cosmic rays.

Since the process of generating truly random sequences of symbols is very difficult, another alternative was developed (Pseudo Random Number Generation)

B. *Pseudo Random Number Generation*

A pseudo-random generator (PRNG) is defined in [10] as a polynomial time computable function g that stretches a short random string x into a long string $g(x)$ that looks like a random string to any feasible algorithm that is allowed to examine $g(x)$. In other words, a PRNG must produce an output that is difficult, if not impossible, to distinguish from a true random source.

A pseudorandom sequence consists of digits that are statistically random up to a specific length, but then repeat themselves exactly. Such pseudorandom sequences can substitute for random sequences if the length of the sequence used is shorter than the length of the cycle in the pseudorandom sequence. The PRN sequences can be generated by many means [9], including:

(a)Linear feedback shift registers, LFSR, (b) Linear congruential generators, (c) Lagged Fibonacci generators, (d) Chebyshev mixing generators, (e) Cellular automata generators, (f) System theoretic approaches, (g) Complexity theoretic approaches, (h)Information-theoretic approaches, (i) Nonlinear shift registers (NLSR), and (j) Deterministic and stochastic chaotic generators.

Pseudorandom sequences have certain advantages over truly random sequences. One of the advantages is the sequence repeatability from one experiment to another. That is, on each successive run, an identical sequence is generated if the seed is the same, while different sequence is produced for different seeds. Thus, different researchers may produce the same results. Another advantage is that a hardware implementation has both smaller area and higher speed [9].

For cryptographic purposes, the seeding of the PRNG is the most critical part of generating a good random source for information derivation. Another way to look at it is to consider that a PRNG utilized for cryptographic purposes must have a high degree of entropy. Where, entropy is a measure of uncertainty. The higher the degree of entropy in a PRNG, the more uncertainty in the prediction of the output produced by that PRNG. As stated in [11], the more entropy we have in an event, the more random the event is expected to be.

In fact, to be of practical use for cryptography, a good pseudo-random sequence is required to satisfy the following properties [8]:

1) The period should be very long, $\approx 10^{50}$ as minimum.
2) The sequence should be easy to generate, for fast encryption.
3) The cryptosystem based on the sequence should be cryptographically secure against chosen plain text attack.

III. Shift Registers

This section introduces an overview of Linear and non-linear Feedback Shift Registers.

A. *Linear Feedback Shift Registers*

The pseudo-random bit sequences can be generated using LFSRs. A common internal structure of LFSR consists of: D flip-flops that form a simple shift register and a number of feedback loops collected into XOR gate [12]. Advantages of LFSRs include the ease of implementation, simplicity, speed, and the ability to generate a maximal cycle sequence with the same uniform statistical distribution of 0's and 1's as in a truly random sequence. The main disadvantage of LFSRs is their linearity, leading to a relatively easier cryptanalysis. A common solution to this problem in LFSR-based stream ciphers is to feed the outputs of several parallel LFSRs into a non-linear Boolean function to form a combination generator. The combining function has to be carefully selected to ensure the security of the resulting scheme against certain attacks, such as correlation attacks. Other approaches combine several bits from the LFSR state using a non-linear function, or use the irregular clocking of the LFSR. As a known example of LFSR-based stream ciphers is the A5/1 stream cipher which is used to provide over-the-air communication privacy in the GSM cellular technology [8].

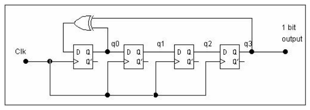

Fig. 1. Four-bit linear feedback shift register

Figure 1 is a four bit linear feedback shift register (LFSR) pseudo random number generator. A LFSR generator, with N bits has a maximum repeating length of $2^N - 1$ cycles [13]. Therefore, the four bit generator has a maximum repeating length of fifteen. The LFSR is initialized with any four bit sequence except all zeroes. If the registers are initialized with zeroes the output will always be zero. If the registers are initialized with 1010 for reg_0, reg_1, reg_2, reg_3, the output sequence, first to last bit out, is 0101100100011110 at which point the sequence will repeat it self.

The input to the first register is the output of an XOR gate whose inputs (or taps) are q_0 and q_3. The tap choice is important to obtain the maximum cycle length. The taps are determined by a recurrence equation [13]

$$X_n = a_1(X_{n-1}) \oplus a_2(X_{n-2}) \oplus \ldots \oplus a_m(X_{n-m})$$

The constants a_is are a predetermined values of zero or one [13]. Many papers, textbooks and application notes contain tables listing taps that produce the maximum cycle length and prevent the LFSR from entering a state with the registers all containing zeros. A LFSR can easily be implemented in digital hardware using registers and an XOR gate. It is better to seed the LFSR with an initial value that comes from a physical or secret entropy source depending on the intended use of the generator. The LFSR is not recommended as a PRNG in software because it has poor random qualities compared to the lagged Fibonacci generator [13], but it much easier to implement in hardware than most of the software generators. The LFSR is basically the Fibonacci generator using the exclusive OR operation. However, a widely studied type of key stream generators (KSG) consists of several LFSRs that are combined by a nonlinear Boolean function.

Mainly LFSRs with primitive characteristics polynomials produce maximum length sequences. It has been pointed out in [14] that the basic requirements for any KSG are: *large period* and *linear complexity*, which is the length of the shortest LFSR. These requirements can be optimized by investigating the minimal polynomial of the key stream. And the minimal polynomial depends on the feedback function and the initial states of the shift registers.

According to the literature, linear complexity of the key stream, randomness and correlation-immune attacks are of great importance. It is possible that sequences with a high linear complexity have a very bad “linear complexity stability”, i.e. after changing a few bits of the original sequence, its linear complexity decreases or increases in a very fast manner [15]. At this point, non-linear feedback shift registers NLFSR can be used to break the linearity problem.

B. Non-Linear Feedback Shift Registers

A Non-linear Feedback Shift Register (FSR) is a generalization of a LFSR where the feedback function is any Boolean function f of n variables. It works in the same way as a LFSR, but the contents of n stages are calculated according to a non-linear function [16]. To illustrate the principle of NLFSR, consider the NLFSR shown in Figure 2. It is defined by the feedback function F(X_0, X_1,…,X_8) = $X_0 \oplus X_7 \oplus (X_1 X_2)$.

IV. Testing for Randomness

The National Institute of Standards and Technology (NIST) has assembled a statistical test suite for true random and pseudo random number generators intended for cryptographic applications [13, 17]. The NIST test suite has 15 statistical

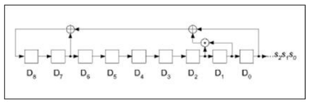

Fig. 2. One-bit implementation of a NLFSR

tests to test a binary sequence of arbitrary length. Each of the 15 tests produces a statistical value that is used to calculate a p-value. The p-value is a quantitative value that compares a perfectly random sequence with the sequence being tested. A p-value equal to one means that the sequence produced by a generator has complete randomness and a sequence having a p-value of zero is entirely non-random.

The NIST test suite consists of the following 15 tests: Frequency test; which tests the uniformity of the bits produced; a test for frequency of bits within an M-bits size block; a Runs test to determine whether the transition between ones and zeros is too fast or too slow; a test for the longest run of ones in a particular size block is the expected length; a random binary matrix test which tests subsets of the sequence for linear independence from other subsets of the sequence; a spectral test to test for periodicity of the sequence; a non-overlapping template matching test; an overlapping template matching test; Maurer’s Universal Statistical test which tests the sequence for compression without loss of information; the linear complexity test; the serial test; the approximate entropy test; the cumulative sums test; the random excursions test; and finally, the random excursions variant test. The effectiveness of the NIST test suite was determined using generators that were known to be good and bad [13]. In this work, randomness testing has been considered in order to enhance our results. More information about some of these tests are shown below.

1) Frequency Test: This test is based on counting the number of ones in the input bit stream. In other words, the distribution of numbers should be uniform [17].

$$\text{Frequency test} = \frac{(n_0 - n_1)^2}{2}$$

2) Serial Test: This test counts the number of turning points (peaks and troughs) in a sequence. The hypothesis is that the proportions of each of all pairs of random numbers are equal.

$$\text{Serial test} = \frac{4(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2)}{n-1} - \frac{2(n_0^2 + n_1^2)}{n+1}$$

3) Run Test: The basic principle behind this test is to look for a run of 0 or 1 in a sequence. The runs test looks at a sequence to see if the oscillations between the zeros and ones are too fast or too slow. The test statistic can be calculated by first counting the length (i) of each run for both zeros (G) and ones (B). A sequence of length n will have an expected number of runs (e_i) for each

length i. The following formulas show how to calculate e_i and the test characteristic [15].

$$X = \sum_{i=1}^{k} \frac{(B_i-e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i-e_i)^2}{e_i}$$
$$e_i = \frac{(n-i+3)}{2^{i+2}}$$

Also, the run test as defined in [9] for a sequence of length K is:

$$\lim_{K\to\infty}\left[\frac{K(1,L_m)}{K(1,L_m+1)}\right] = 2,\ m = 1, 2, \ldots, (\tfrac{k}{2}) - 1$$

where K(1,L_m) is the number of runs of consecutive Is in groups of L_m bits each, and K(1, L_m+1) is the number of runs of consecutive ones in groups of L_m+1 bits each. Each run (a group of consecutive is) is flanked by 0s on both sides. This criterion stems from the observation that a perfectly random sequence has 1-bit runs of length L_m twice as probable as 1-bit runs of length L_m+1.

V. The Proposed Pseudo Random Number Generation Schemes

In this work, several Pseudo Random Number Generators based on 80-bits LFSR and NLFSR with different feedback functions and taking in consideration the Grain stream approach [18] were proposed. Then, the proposed PRN generators were implemented by describing them using hardware description language (VHDL) and simulated for functional correctness using Mentor Graphics tools (ModelSim).

A. LFSR and NLFSRs

For thorough evaluation, LFSR design with 80 bits key stream was implemented. The function considered for the LFSR is:

L(X_0, …, X_{79}) = $X_{62} \oplus X_{51} \oplus X_{38} \oplus X_{23} \oplus X_{13} \oplus X_0$

Then, and due to their importance, four NLFSRs of lengths 24, 27, 32 and 80 bits were implemented. The feedback functions of the four NLFSRs are:

N(X_0, …, X_{23}) = $X_0 \oplus X_1 \oplus X_3 \oplus X_5 \oplus X_6 \oplus X_7 \oplus X_9 \oplus X_{12} \oplus X_{14} \oplus X_{15} \oplus X_{17} \oplus X_{18} \oplus X_{22} \oplus (X_1X_6) \oplus (X_4X_{13}) \oplus (X_8X_{16}) \oplus (X_{12}X_{15}) \oplus (X_5X_{11}X_{14}) \oplus (X_1X_4X_{11}X_{15}) \oplus (X_2X_5X_8X_{10})$;

N(X_0, …, X_{26}) = $X_0 \oplus X_1 \oplus X_2 \oplus X_7 \oplus X_{15} \oplus X_{17} \oplus X_{19} \oplus X_{20} \oplus X_{22} \oplus X_{26} \oplus (X_9\ X_{17}) \oplus (X_{10}\ X_{18}) \oplus (X_{11}X_{14}) \oplus (X_{12}X_{13}) \oplus (X_5X_{14}X_{19}) \oplus (X_6X_{10}X_{12}) \oplus (X_6X_9X_{17}X_{18}) \oplus (X_{10}X_{12}X_{19}X_{20})$;

N(X_0, …, X_{31}) = $X_0 \oplus X_2 \oplus X_6 \oplus X_7 \oplus X_{12} \oplus X_{17} \oplus X_{20} \oplus X_{27} \oplus X_{30} \oplus (X_3\ X_9) \oplus (X_{12}\ X_{15}) \oplus (X_4X_5X_{16})$

N(X_0, …, X_{79}) = $\oplus\ X_{62} \oplus X_{60} \oplus X_{52} \oplus X_{45} \oplus X_{37} \oplus X_{33} \oplus X_{28} \oplus X_{21} \oplus X_{14} \oplus X_9 \oplus X_0 \oplus (X_{63}X_{60}) \oplus (X_{37}X_{33}) \oplus (X_{15}X_9) \oplus (X_{60}X_{52}X_{45}) \oplus (X_{33}X_{28}X_{21}) \oplus (X_{63}X_{28}X_{45}X_9) \oplus (X_{60}X_{52}X_{37}X_{33}) \oplus (X_{63}X_{60}X_{21}X_{15}) \oplus (X_{63}X_{60}X_{52}X_{45}X_{37}) \oplus (X_{33}X_{28}X_{21}X_{15}X_9) \oplus (X_{52}X_{45}X_{37}X_{33}X_{28}X_{21})$

B. Grain

Finally, we implement combination of LFSR and NLFSR taking into account the Grain approach that was presented in [18].

The new stream cipher, Grain, was introduced in [18], which is shown in Figure 3. The construction is based on two shift registers, one with linear feedback and one with nonlinear feedback, and a nonlinear filter function. The key size is 80 bits and no attack with complexity better than exhaustive key search has been identified. Grain is a bit oriented stream cipher producing 1 bit/clock in its simplest implementation.

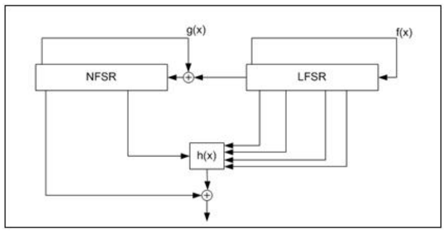

Fig. 3. The cipher

The design of a cipher needs to focus on some specific properties. It is not possible to have a design that is perfect for all purposes i.e., processors of all word lengths, all hardware applications, all memory constraints, etc. Grain is designed to be very small in hardware, using as few gates as possible, while maintaining high security. The cipher is intended to be used in environments where gate count, power consumption and memory needs to be very small [18]. While Grain is still possible to be used in general application software, there are several ciphers that are designed with software efficiency in mind and thus are more appropriate when high speed in software is required. Because of this it does not make sense to compare the software performance of Grain with other ciphers.

Due to the statistical properties of maximum-length LFSR sequences, the bits in the LFSR are (almost) exactly balanced. This may not be the case for a NLFSR when it is driven autonomously. However, as the feedback $g(x)$ is xored with a LFSR-state, the bits in the NLFSR are balanced. Where:

f(x) = $x_{62} \oplus x_{51} \oplus x_{38} \oplus x_{23} \oplus x_{13} \oplus x_0$;

g(x) = $x_0 \oplus x_{62} \oplus x_{60} \oplus x_{52} \oplus x_{45} \oplus x_{37} \oplus x_{33} \oplus x_{28} \oplus x_{21} \oplus x_{14} \oplus x_9 \oplus x_0 \oplus (x_{63}x_{60}) \oplus (x_{37}x_{33}) \oplus (x_{15}x_9) \oplus (x_{60}x_{52}x_{45}) \oplus (x_{33}x_{28}x_{21}) \oplus (x_{63}x_{28}x_{45}x_9) \oplus (x_{60}x_{52}x_{37}x_{33}) \oplus (x_{63}x_{60}x_{21}x_{15}) \oplus (x_{63}x_{60}x_{52}x_{45}x_{37}) \oplus (x_{33}x_{28}x_{21}x_{15}x_9) \oplus (x_{52}x_{45}x_{37}x_{33}x_{28}x_{21})$

Moreover, recall that g is a balanced function. Therefore, the bits in the NLFSR may be assumed to be uncorrelated to the LFSR bits. The function h is chosen to be correlation-immune of first order. This does not preclude that there are correlations of the output of $h(x)$ to sums of inputs. As one input comes from the NLFSR and as $h(x)$ is xored with a state bit of the NLFSR, correlations of the output of the generator to sums of LFSR-bits will be so small that they will not be exploitable by fast correlation attacks.

VI. Results, Evaluation and Comparisons

As mentioned earlier, the proposed schemes were described in VHDL and simulated using ModelSim. The generated sequences have been tested for randomness properties according to three of the statistical test suite published by the National Institute of Standards and Technology (NIST) [11]. These tests are: the frequency test, the Serial test and the Run test. The resulted key-streams are shown in the Table below.

A. Analysis of Random Sequences

There were many ways that can be found in literature to evaluate the security of stream ciphers. First, mathematical analysis to test the period, and linear complexity, and security against correlation attacks. The second way, is pseudo random testing that includes statistical tests, linear complexity, Ziv-Lempel Complexity and Maximum Order Complexity [9].

Thus, some statistical properties can be exploited to evaluate and compare in a reasonable manner the strength of our results. Table 1 shows the analysis results of the above generated key streams. For example, the sequence LFSR key stream contains 33 randomly distributing 1's among 47 zeros. It has σ-runs of varying length L, denoted by K(σ, L). For example, the 1-runs are K(1,1)= 7, K(1,2)=5, K(1,3)=4, K(1,4)=1. The 0-runs include K(0,1)=6, K(1,2)=2, K(0,3)=5, K(0,4)=2, K(0,5)=0, K(0,+6)=2. From both the balance of 1s and 0s that is close to 1.5, the ratio of runs (e.g., K(1,1)/K(1,2)) that is close to 1.4, and we conclude that is a random sequence.

Regarding NLFSRs, we have changed both the length and the feedback function. It appears through statistical tests that NLFSR is better than LFSR. Specifically, the distribution of 0's and 1's is more uniform and the proportions of each pair of 00, 01, 10 and 11 are equal compared to LFSR as shown in Table 1. On the other hand, the Grain stream introduces much better statistical characteristics than LFSR, where they have the same length. From practical point of view, the Grain would be much better than alone NLFSR, since the Grain benefits from the LFSR output linearity.

On the other hand, if we consider NLFSR-80 bit key stream, the 1-runs are K(1,1)= 17, K(1,2)=9, the ratio of runs (K(1,1)/K(1,2) is close to 2, which is better than LFSR. In general, it appears that NLFSRs and Grain are significantly much better than LFSR in both the frequency and serial tests.

Table 1: Evaluation of frequency test (Test 1) and serial test (Test 2)

Approach	n_0	n_1	n_{00}	n_{01}	n_{10}	n_{11}	Test 1	Test 2
LFSR 80	47	33	30	17	16	16	2.45	4.7
NLFSR-24	12	12	7	5	4	6	0	1.13
NLFSR-27	15	13	6	9	8	4	0.148	2.16
NLFSR-32	17	15	11	6	6	8	0.125	2.0
NLFSR-80	38	42	16	21	22	20	0.2	0.84
Grain	42	38	23	19	17	19	0.2	1.2

VII. Conclusion

This work presented implementations of different RNGs; LFSR, NLFSRs and Grain using ModelSim simulation tool for hardware described designs using VHDL. According to NIST tests, the results proved that NLFSRs produce random numbers that are much better than LFSR generators. Our proposed Grain stream cipher that combines 80-bit LFSR and 80-bit NLFSR achieved approximately the same improvements.

In other words, the efficiency of the proposed NLFSR schemes are verified by comparing it against LFSR based on frequency test and bit serial test pattern generators. In addition, it has been proved that the usage of two sources of pseudorandom sequence bits to create a third source of pseudorandom sequence bits produces better quality than the original sources.

References

[1] Gaston E. Barberis, "Non-periodic pseudo-random numbers used in Monte Carlo calculations", *Physica* B 398, 468-471, 2007.

[2] Yas Abbas Alsultanny, "Random-bit sequence generation from image data", *Image and Vision Computing*, 26, 592-601, 2008.

[3] Yue Hu *et al.*, "A true random number generator based on mousemovement and chaotic cryptography", *Chaos, Solitons and Fractals*, 2007.

[4] Ali Kanso and Nejib Smaoui, "Logistic chaotic maps for binary numbers generations",*Chaos, Solitons and Fractals*, 2007. doi:10.1016/j.chaos.2007.10.049.

[5] Raul Gonzalo Diaz, "A New Public-Key Cryptosystem Family Based on Feedback Shift Registers", *IEEE*, 1999.

[6] Zibin Dai, "Design and Implementation of A High-speed Reconfigurable Feedback Shift Register", *IEEE*, 2008.

[7] Said E. El- Khamy *et al.*, "The FBG Stream Cipher", *IEEE*, 2007.

[8] Elena Dubrova, *et al.*, "On Analysis and Synthesis of (n,k)-Non-Linear Feedback Shift Registers", *EDAA*, 2008.

[9] W. Kinsner, "Single-Scale Measures For Randomness and Complexity", *Proc. 6th IEEE Int. Conf. on Cognitive Informatics (ICCI'07)* , 554-568, 2007.

[10] http://www.nist.gov/dads/HTML/pseudorandomNumberGen.html

[11] "Federal Information Processing Standards Publication 186-2", Specifications for the Digital Signature Standard, 2000, National Institute of Standards and Technology.

[12] A. Jutman, *et al.*,"A Tool for Advanced Learning of LFSR-Based Testing Principles", *IEEE*, 2006.

[13] Jennifer L. Brady. "Limitations of a True Random Number Generator in a Field Programmable Gate Array", [Thesis]. *Air Force Institute of Technology, Air University*. Dec. 2007.

[14] Bemdt M. Gammel et al., "An NLFSR-Based Stream Cipher", Infineon Technologies AG, Munich, Germany, *IEEE*, 2006.

[15] Ral Gonzalo, *et al.*, "Non-Linear Feedback Shift Registers with Maximum Period", *Technical Report*, 1997.

[16] John Mattson, "Stream Cipher Design" [Thesis]. *KTH Computer Science and Communication*, Stockholm, Sweden 2006.

[17] Andrew Weigl, Walter Anheier, "Hardware Comparison of Seven Random Number Generator Tests for Smart Cards", *Technical Report, University of Bremen, Germany*, 2001.

[18] Martin Hell *et al.*, "Grain - A Stream Cipher for Constrained Environments", *IEEE*, 2007.