Primitive polynomials selection method for pseudo-random number generator

I V Anikin, Kh Alnajjar

Kazan National Research Technical University named after A.N.Tupolev-KAI, Kazan, 420111, Russia.

e-mail: anikinigor777@mail.ru, khkazan@mail.ru

Abstract. In this paper we suggested the method for primitive polynomials selection of special type. This kind of polynomials can be efficiently used as a characteristic polynomials for linear feedback shift registers in pseudo-random number generators. The proposed method consists of two basic steps: finding minimum-cost irreducible polynomials of the desired degree and applying primitivity tests to get the primitive ones. Finally two primitive polynomials, which was found by the proposed method, used in pseudorandom number generator based on fuzzy logic (FRNG) which had been suggested before by the authors. The sequences generated by new version of FRNG have low correlation magnitude, high linear complexity, less power consumption, is more balanced and have better statistical properties.

1. Introduction

Nowadays high quality pseudorandom numbers have a critical importance to many scientific applications. Pseudorandom number generators based on linear feedback shift registers (LFSRs) [1] are among the fastest long-period generators currently available. They require less hardware for implementation, have more higher speed of operations and produce pseudorandom sequences with good statistical properties. In papers [2,3] we suggested new type of pseudorandom number generator based on fuzzy logic - FRNG. The structure of this generator is presented on Figure 1.

![Figure 1. General structure of proposed FRNG.](image-url)
This generator involves several LFSRs, buffers and fuzzy non-linear function with two fuzzy variables $f_0$ (number of ones) and $|f_1 - f_2|$ (the difference between number of two consecutive ones (0110) and number of two consecutive zeros (1001)), and a group of fuzzy if-then rules.

In [3] we evaluated the quality of suggested generator and showed that it passed basic statistical tests from NIST and Diehard bundles. Moreover, this generator showed better quality than some other well-known generators. Nevertheless, it is extremely important to investigate crucial parameters of the suggested generator and choose the best ones to improve the quality and the efficiency of generated pseudo-random numbers sequences. One of the basic parameters are characteristic polynomials of LFSRs which are used in FRNG.

It's well known that a binary sequence generated by LFSR possesses good statistical properties if its characteristic polynomial is primitive. For example, the period $T$ of such sequence (is called M-sequence) is $2^n-1$, where $n$ is the degree of the polynomial. M-sequences are used for obtaining uniformly distributed random numbers [4] and they widely used in practice, for example in modeling or cryptography. The quality of M-sequences grows with increasing the degree $n$ of the characteristic polynomial of LFSR, so it is very important in practice to obtain primitive polynomials with big degrees.

There are several published tables with primitive binary polynomials. Watson in [5] gives one primitive of degree $n$ for every $n<100$. Stahnke in [6] gives a primitive with a minimum number of nonzero coefficients (trinomial or pentanomial) for every $n\leq168$. One primitive pentanomial for every degree $M_j$, $8 \leq j \leq 27$, is also presented in [7] (here $M_j$ denotes the $j$th Mersenne exponent, the prime for which $(2^{M_j}-1)$ is also prime). 5,9,17-weight primitive polynomials of degree up to 800 over GF(2) are presented in [8].

Aside from uniform distribution, good $F_2$-linear generators are also required to have characteristic polynomials $P(x)$ whose number of non-zero coefficients is not too far from the half of degree, i.e., approximately $n/2$ [9]. In particular, generators for which $P(x)$ is a trinomial or a pentanomial, which have been widely used in the past, should be avoided. It is known, that these generators have different statistical weaknesses [10].

In this paper we propose a method of finding good primitive binary polynomials. These polynomials have good statistical properties, high diffusion capacity (due to relatively high number of non-zero coefficients) and low power consumption.

The proposed method includes two main stages:

- Finding minimum-cost irreducible polynomials of the desired degree.
- Applying primitivity tests to find primitive polynomials.

This paper is organized as follows: in the section 2 we describe the step related with finding minimum-cost irreducible polynomials of the desired degree, in the section 3 we describe the step related with applying primitivity tests to get only primitive polynomials, in the section 4 we describe some experimental results, in Conclusion we give the basic results of the research.

2. Finding minimum-cost irreducible polynomials of the desired degree

It was proved in [8], that characteristic polynomial $f(x)$ of special type can be used to construct an $n$-stage standard LFSR with minimum-cost (concerning the number of 2-input XOR gates needed to implement the LFSR). These polynomials can be described by the general expression (1):

$$f(x) = (1 + x^{b_1})(1 + x^{b_2})...(1 + x^{b_m}) + x^n$$  \hspace{2cm} (1)

Parameters of these polynomials should satisfy the following conditions (2):

$$b_1 \geq 1, b_1 < b_2, (b_1 + b_2) < b_3, \cdots, (b_1 + b_2 + \cdots + b_{m-1}) < b_m, (b_1 + b_2 + \cdots + b_m) < n$$  \hspace{2cm} (2)

We will use this type of polynomials to fit the requirements of high efficiency regarding power consumption which directly depends on the number of 2-input XOR gates needed to implement it in hardware (In our case it's equal to $m+1$). These types of polynomials have also a high diffusion capacity which depends on the number of non-zero coefficients, which should be not too far from the half of degree $n/2$. This number is defined as $t = 2^m + 1$. For example if $n = 11$ then the most suitable
choice of m is 2 where we can get a polynomial with \( t = 2^2 + 1 = 5 \) non-zero coefficients. We can take \( b_1 = 1, b_2 = 5 \) to get the following irreducible polynomial as example:

\[
f(x) = (1 + x^1)(1 + x^5) + x^{11}
\]

We can select all the irreducible polynomials of degree \( n = 11 \) and satisfy the conditions defined by (2) in this case \( (b_1 \geq 1, b_2 > b_1, b_1 + b_2 < 11) \). Table 1 contains all possible variants that fit these requirements:

**Table 1.** List of all 5-wights (t=5) irreducible polynomials of the degree \( n=11 \).

| \( b_1 \) | \( b_2 \) | \( f(x) \) |
|---|---|---|
| 1 | 5 | \((1 + x^1)(1 + x^5) + x^{11}\) |
| 1 | 6 | \((1 + x^1)(1 + x^6) + x^{11}\) |
| 1 | 7 | \((1 + x^1)(1 + x^7) + x^{11}\) |
| 1 | 8 | \((1 + x^1)(1 + x^8) + x^{11}\) |
| 2 | 3 | \((1 + x^2)(1 + x^3) + x^{11}\) |
| 2 | 6 | \((1 + x^2)(1 + x^6) + x^{11}\) |
| 2 | 7 | \((1 + x^2)(1 + x^7) + x^{11}\) |
| 3 | 5 | \((1 + x^3)(1 + x^5) + x^{11}\) |
| 3 | 6 | \((1 + x^3)(1 + x^6) + x^{11}\) |
| 3 | 7 | \((1 + x^3)(1 + x^7) + x^{11}\) |

In the previous example \( (n=11, m=2) \) the complete list of irreducible polynomials is not too long, which is not the general case. The number of polynomials will grow very fast with increasing the parameters \( n \) and \( m \). For example if \( m=5 \) and the desired degree is 67 we will get a list of 147 irreducible polynomials with \( t=33 \) non-zero coefficients.

In most practical applications we need a polynomials with high quality. Therefore there is a need to find primitive polynomials with the degree as high as possible. Table 2 contains some values of \( m \), the corresponding number of non-zero coefficients \( t \), and the range of degrees which should be used with them.

**Table 2.** Values of \( m \) with corresponding number of non-zero coefficients \( t \) and the suitable ranges of degrees \( n \) of polynomials.

| \( m \) | \( t \) | \( n \) |
|---|---|---|
| 4 | 17 | 35-66 |
| 5 | 33 | 67-130 |
| 6 | 65 | 131-258 |
| 7 | 129 | 259-514 |
| 8 | 257 | 515-1026 |

We used *Mathematica* software to write a program which find all irreducible polynomials of type (1) with \( 2^m+1 \) non-zero coefficients for desired degree. This program enumerates all the possible values of the parameters \( (b_1, b_2, \ldots, b_m) \) which satisfy all the previous mentioned conditions (2). The result is a list of numbers \( (b_1, b_2, \ldots, b_m) \) ending with the degree of the polynomials. For example if \( m=5, \ n=67 \), one of the resulting element will be \( (2, 3, 7, 16, 33, 67) \) that means:
\[ f(x) = (1 + x^2)(1 + x^3)(1 + x^7)(1 + x^{16})(1 + x^{33}) + x^{67} \]

We can expand the expression and get the irreducible polynomial with \( r = 33 \) terms:

\[
\begin{align*}
    f(x) &= 1 + x^2 + x^3 + x^5 + x^7 + x^9 + x^{10} + x^{12} + x^{16} + x^{18} + x^{19} + x^{21} + x^{23} + x^{25} + x^{26} \\
    &\quad + x^{28} + x^{33} + x^{35} + x^{36} + x^{38} + x^{40} + x^{42} + x^{43} + x^{45} + x^{49} + x^{51} + x^{52} + x^{54} \\
    &\quad + x^{56} + x^{58} + x^{59} + x^{61} + x^{67}
\end{align*}
\]

Finally, at the end of this step we obtain the full list of irreducible polynomials which satisfy the conditions (2). Then we need to select only the primitive ones from them, and that will be done in the next step.

3. Applying primitivity tests and finding primitive polynomials from the list

After generating the lists of irreducible polynomials over \( \text{GF}(2) \) we should test them to get only the primitive ones \[11\]. The primitivity test of a given polynomial \( f \) is effectively performed with using the following set of conditions \[12\]:

- \( f(0) = f(1) = 1 \)
- \( \min \{ k : f(x^k - x) = n \} \)
- for all primes \( p \)
  
  \[
  p^2 - 1, \quad x^{(x+1)/p} \neq 1 \pmod{p}.
  \]

The first condition eliminates polynomials which is divisible by \( x \) and \( x+1 \). In our case the polynomial \( f \) has the type defined by (1) and this condition is fulfilled automatically.

To check if the polynomial \( f \) satisfies the second condition, it is necessary to calculate \( n \) residues \( x^{2^k} - x \mod f \). The total number of elementary operations over \( \text{GF}(2) \) which should be done to test the condition (2) is bounded by \( O(n^2) \), which is big, taking into account the number of polynomials that is needed to be checked. The problem is solved in \[14\] where condition (2) is modified by checking two subconditions:

- \( \gcd(f, f') = 1 \)
- \( \gcd(f, x^{2^k} - x) = 1, \quad 2 \leq k \leq 12. \)

Here \( \gcd(f, g) = 1 \) is the greatest common divisor of the polynomials \( f \) and \( g \) \[13\].

The computing complexity of factorization of \( 2^n - 1 \), which is needed in the third condition, is very high. This makes it inefficient and unreasonable to include this checking in the primitivity test and it makes the testing process extremely slow. Fortunately we can avoid it according to the famous Cunningham project \[14\]. We build a much faster method by storing the factors in database depending on \[14\]. Furthermore, if we choose the degree \( n \) to be a prime, then the number of factors of \( 2^n - 1 \) will be very small, so the speed of the primitivity test will be very high. For example, when \( n = 63 \) the factors of \( 2^{63} - 1 \) is \{7,2,73,1,127,1,{337,1}, {92737,1}, {649657,1}\} we have to check six cases, but with \( n = 67 \) the factors of \( 2^{67} - 1 \) will be \{193707721,1,761838257287,1\}. So we have to check only two polynomials. The complexity of this primitivity check is \( O(F \cdot n^2) \), where \( F \) is the number of distinct prime factors of \( 2^n - 1 \).

In our algorithm we listed the prime factors of \( 2^n - 1 \) for all primes \( n \) between 35 and 1026 in a table according to \[15\]. Then we created the program in Mathematica software \[16\] to generate the lists of high efficient irreducible polynomials of the desired degree with the suitable \( m \) (see Table II). Finally we wrote the algorithm and program for primitivity test to pick up only primitive polynomials from the created list of irreducible polynomials.

4. Experimental results
Applying the suggested method we found all primitive polynomials with prime degree \( n \) between 35 and 1026. Some examples are represented in the Table 3.

| \( n \) | \( t \) | \( f(x) \) |
|-------|-------|---------|
| 37    | 17    | (3, 4, 9, 17, 37) |
| 61    | 17    | (2, 3, 15, 39, 61) |
| 67    | 33    | (1, 2, 4, 9, 41, 67) |
| 89    | 33    | (1, 5, 10, 17, 39, 89) |
| 149   | 65    | (3, 5, 9, 19, 37, 75, 149) |
| 251   | 65    | (1, 2, 4, 14, 23, 141, 251) |
| 313   | 129   | (2, 3, 6, 12, 24, 65, 169, 313) |
| 509   | 129   | (3, 6, 13, 27, 54, 119, 266, 509) |
| 983   | 257   | (2, 5, 12, 28, 58, 118, 243, 512, 983) |

In order to increase the efficiency of the suggested in [2,3] FRNG, we replaced two primitive characteristic polynomials (3) currently used in FRNG by two primitive ones (4) selected from the obtained lists with using the proposed method.

\[
P_1(x) = 1 + x^{38} + x^{42} + x^{46} + x^{42} \\
P_2(x) = 1 + x^{90} + x^{97} \\P_1(x) = (1 + x^7)(1 + x^7)(1 + x^{10})(1 + x^{17})(1 + x^{30}) + x^89 \\
P_2(x) = (1 + x^7)(1 + x^7)(1 + x^{20})(1 + x^{53}) + x^97
\] (3)

Finally we evaluated the quality of our FRNG (with two selected primitive polynomials (4)) with using NIST [17] and Diehard [18] bundles. The new FRNG successfully passed all randomness tests. Furthermore the new version of FRNG became more secure against algebraic attacks due to using special type of polynomials which have high diffusion capacity and high linear complexity, which increase the immunity of the proposed generator against correlation attacks.

5. Conclusion
Finally, we can conclude that the proposed method of selection primitive polynomials is very useful for the field of pseudo-random numbers generators. Selected primitive polynomials increase the efficiency of pseudorandom numbers generators. As a result the sequences generated by new version of FRNG will have low correlation magnitude, high linear complexity, less power consumption, will be more balanced, they will have good statistical properties. It makes the resulting FRNG more suitable for various applications such as modeling, telecommunication, cryptography, authentication, etc.

6. References
[1] Lewis T G and Payne W H 1973 Generalized Feedback Shift Register Pseudorandom Number Algorithm Journal of the ACM (JACM) 20 456-468.
[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] Tausworthe R C 1965 Random numbers generated by linear recurrence modulo two *Math. Comp.* **19** 201-209.

[5] Watson J 1962 Primitive polynomials (mod 2) *Math. Comp.* **16** 368-369.

[6] Stahnke W 1973 Primitive binary polynomials *Math. Comp.* **27** 977-980.

[7] Kurita Y and Matsumoto M 1991 Primitive t-nomials (t = 3; 5) over GF(2) whose degree is a Mersenne exponent ≤ 44497 *Math. Comp.* **56** 817-821.

[8] Wang Lang-Tereng, Touba Nur A, Brent Richard P, and Wang Hui 2011 On Designing Transformed Linear Feedback Shift Registers with Minimum Hardware Cost *UT-CERC-12-03*.

[9] Wang D and Compagner A 1993 On the use of reducible polynomials as random number generators *Mathematics of Computation* **60** 363–374.

[10] Lindholm J H 1968 An analysis of the pseudo-randomness properties of subsequences of long m-sequence *IEEE Transactions on Information Theory* **IT-14 (4)** 569–576.

[11] Watson E J Primitive polynomials (mod 2) *Mathematics of Computation* **16(79)** 368-368.

[12] Lidl R and Niederreiter H 1983 Finite Fields, Encyclopedia *Math. Appl* **20**.

[13] Cohen E Arithmetical functions of a greatest common divisor *Proceedings of the American Mathematical Society* **11(2)** 164-171.

[14] Heringa J R, Blote H W and Compagner 1992 A New Primitive trinomials of Mersenne-exponent degrees for random number generation *International Journal of Modern Physics* **3** 561-564.

[15] Brillhart J, Lehmer D H, Selfridge J L, Tuckerman B, and Wagstaf S S 1988 Factorization of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers **22 Amer.Math. Soc.**

[16] [https://www.wolfram.com/mathematica/](https://www.wolfram.com/mathematica/)

[17] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J and Vo S 2010 A statistical test suite for random and pseudorandom number generators for cryptographic applications *NIST Special Publication 800-22, Revision 1a*. 