Annales Mathematicae et Informaticae
56 (2022) pp. 36–46
DOI: https://doi.org/10.33039/ami.2022.12.012
URL: https://ami.uni-eszterhazy.hu

Using irreducible polynomials for random number generation

Tamás Herendi[a*†], **Sándor Roland Major**[b*]

[a]Department of Computer Science
University of Debrecen
Debrecen, Hungary
herendi.tamas@inf.unideb.hu

[b]Department of Information Technology
University of Debrecen
Debrecen, Hungary
major.sandor@inf.unideb.hu

Abstract. A method is presented for generating random numbers with uniform distribution using linear recurrence sequences with very large period lengths. This method requires an irreducible polynomial modulo 2 to define the sequence. A suitable method for generating an infinite number of such polynomials is presented. The polynomials generated in this way can have an arbitrarily large degree, and a large enough order to make them suitable for practical applications.

Keywords: irreducible polynomials, finite fields, random number generation

AMS Subject Classification: 12-08 Computational methods for problems pertaining to field theory

1. Introduction

Pseudorandom number generation (PRNG) is an important component of many practical applications. Generators with different properties are used in a wide

*The author has been partially supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

†The author has been partially supported by the SETIT Project (no. 2018-1.2.1-NKP-2018-00004), which has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme.

Submitted: November 2, 2022
Accepted: December 3, 2022
Published online: December 28, 2022

range of fields, such as simulations [12], Monte Carlo methods [14]. Morover, PRNGs recently play an essential role in many areas of cryptography, for example, key generation, stream ciphers, asymmetric cryptosystems, and authentication protocols [11].

The results presented in this paper relate to an algorithm detailed in [8], showing the construction of uniformly distributed linear recurrence sequences (LRS) modulo powers of 2, with theoretically arbitrarily large period lengths. A modified version of this algorithm is given in Section 3, optimizing it to be less computationally expensive.

2. Theory

The algorithm presented takes an irreducible polynomial over $\mathbb{F}_2$ as input.

Irreducible polynomials over finite fields are used in a wide variety of contexts, not just in pure mathematics and many areas of computer science, but practical applications as well.

In [8], we see a method of constructing linear recurring sequences with extremely long periods used for pseudorandom number generation. The sequence requires an irreducible polynomial to create, the degree of which is directly related to the resulting period length.

In coding theory, creating error correcting codes that can be used to reliably transmit information over noisy channels is a key practical application that can be found in many everyday electronic systems. These codes are almost always connected to the use of polynomials over finite fields. An in-depth discussion can be found in [9].

In cryptography, many encryption protocols use finite fields as their domain. Irreducible polynomials have been used in public key cryptosystems for decades, such as in [4].

As the previous examples show, irreducible polynomials over the finite field $\mathbb{F}_2$ are of special interest.

2.1. Irreducible polynomials

A univariate polynomial over the finite field $\mathbb{F}_2$ is

$$p(x) = \sum_{k=0}^{n} a_k x^k, \;\; a_0, \ldots, a_n \in \mathbb{F}_2 \,.$$

$\mathbb{F}_2[x]$ is the set of all polynomials over $\mathbb{F}_2$. A polynomial $p \in \mathbb{F}_2[x]$ of degree k is irreducible if it has no nontrivial factors over $\mathbb{F}_2$. That is, $p(x) = p_1(x)p_2(x)$ can not hold if $\deg(p_1), \deg(p_2) > 0$. The natural way to prove a polynomial's irreducibility is, therefore, to factor it and show that no such factors can be found.

The first algorithm for factoring a polynomial over a finite field was published by Berlekamp [2]. It is a deterministic algorithm that requires a square-free polynomial, and is well suited for cases where the cardinality of the finite field is small.

Later, the Cantor-Zassenhaus algorithm [3] provided a practical solution even for polynomials over large finite fields. This algorithm is probabilistic in nature. A detailed description of both methods can be found in [13].

Rabin's test [16] provides a very simple algorithm. A polynomial over $\mathbb{F}_2$ is irreducible if and only if:

1. $p(x) \mid x^{2^k} - x$
2. $\forall t_i \;\; \mathrm{GCD}(x^{2^{k/t_i}}, p(x)) = 1,$

where t_i are the prime divisors of k. The test simply computes all $x^{2^{k/t_i}} \bmod p(x)$ polynomials using repeated squaring, and polynomial modulo operations, then uses polynomial GCD to check condition 2.

Ben-Or's test [1] modifies this approach by computing $\mathrm{GCD}(x^{2^i} \bmod p(x), p(x))$ for every $i \in \{1, \ldots, \frac{n}{2}\}$. In practice, this improves average performance when testing random polynomials. A randomly selected polynomial is much more likely to have factors of small degrees than be the product of only large-degree factors. Since Ben-Or's test checks for factors of small degrees first, these polynomials are very quickly eliminated. A comparison between the performance of Rabin's test and Ben-Or's test can be found in [7]. Victor Shoup also published a deterministic irreducibility test in [19], and a probabilistic algorithm is [18].

3. Algorithm for creating LRS

The following algorithm is for constructing uniformly distributed linear recurrence sequences modulo 2^s, with very large period lengths. It is a modified version of the algorithm found in [8]. The version presented here is significantly less computationally expensive than the original, which enables the creation of sequences with larger period lengths.

The reduced time complexity speeds up the process of finding the desired coefficients for the LRS, while the reduced space complexity allows the algorithm to be carried out with significantly larger input parameters. Once the LRS is constructed, using it to generate the pseudorandom number sequence is unchanged compared to the original version.

1. Choose an integer k and find a monic polynomial $q(x) \in \mathbb{Z}[x]$ of degree k, which reduction modulo 2 is irreducible in $\mathbb{F}_2[x]$.
2. Calculate the polynomials $p(x)$ of degree $k+2$ and $p'(x)$ of degree $k+1$ in the following way:

$$p(x) \equiv (x^2 - 1)q(x) \bmod 2 \quad \text{and}$$
$$p'(x) \equiv (x - 1)q(x) \bmod 2,$$

with the coefficients of $p(x)$ and $p'(x)$ in $\{0, -1\}$, except for the leading coefficients.

Calculate the four candidate polynomials:

$$\begin{aligned} p_1(x) &= p(x) \\ p_2(x) &= p(x) - 2 \\ p_3(x) &= p(x) - 2x \\ p_4(x) &= p(x) - 2x - 2 \end{aligned}$$

We remark that the coefficients of the constant and linear terms of the candidate polynomials can be in $\{0, -1, -2, -3\}$.

3. For $i \in \{1, 2, 3, 4\}$, $j \in \{0, 1, \ldots k + 2\}$, let a_{ij} denote the coefficient of x^j in the polynomial $p_i(x)$. Calculate $S_i = \sum_{j=0}^{k+1} -a_{ij}$ for each candidate polynomial. Keep the two candidates that satisfy $S_i \equiv 1 \mod 4$. Denote these two polynomials with c_1 and c_2.

4. Let $\varrho = \mathrm{ord}(q)$ be the order of $q(x)$, i.e., the smallest positive integer such that $q(x) \mid x^\varrho - 1$.

   We need to find the candidate that satisfies $c_i(x) \nmid x^{2\varrho} - 1 \mod 4$. To do this, calculate

   $$r(x) \equiv x^\varrho \bmod (2, p(x)),$$

   where $\mathrm{mod}(2, p(x))$ means calculating the polynomial remainder with $p(x)$ over $\mathbb{F}_2$.

   Then, find the candidate that satisfies

   $$1 \not\equiv r(x)^2 \bmod (4, c_i(x)),$$

   where $\mathrm{mod}(4, c_i(x))$ means calculating the polynomial remainder with $c_i(x)$ over $\mathbb{F}_4$.

   Note that all of the computation in this step can be performed over $\mathbb{F}_2$, with the exception of the last step, which is performed over $\mathbb{F}_4$.

   Denote the candidate that remains by $c(x)$. This is the characteristic polynomial of the linear recurrence sequence we want to create. Let b_j, $j \in \{0, 1, \ldots k + 2\}$ be the coefficient of x^j in $c(x)$. Then, our final recurrence relation is

   $$u_{n+k+2} = -b_{k+1}u_{n+k+1} - b_k u_{n+k} \ldots - b_0 u_n$$

5. Choose initial values for the sequence. Suppose we want s-bit long pseudorandom numbers. Choose random $u_0, u_1, \ldots, u_k \in [0, 2^s - 1]$. Set these values as the initial values of the linear recurrence relation with characteristic polynomial $p'(x)$. Compute the next element of the sequence, u'_{k+1}. Find a random number $u_{k+1} \in [0, 2^s - 1]$ such that $u'_{k+1} \not\equiv u_{k+1} \bmod 2$.

   Set $u_0, u_1, \ldots, u_{k+1}$ as the initial values of the sequence.

The original version of the above algorithm differs in steps 3 and 4. That algorithm requires computation using the companion matrices of the candidates. The companion matrix of $p_i(x)$ is

$$M_{(i)} = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_{i0} & -a_{i1} & \cdots & -a_{ik} & -a_{ik+1} \end{pmatrix}$$

Step 3 calls for finding the two candidates that satisfy $M_{(i)}\bar{1} \equiv \bar{1} \bmod 4$, where $\bar{1}$ is a vector of size $k+2$ with all coordinates equal to 1. It is simple to see that this is equivalent to the step 3 of the above algorithm.

In step 4 of the old algorithm, the final candidate is the one for which $M_{(i)}^{2\varrho} \not\equiv E \pmod 4$ holds, where E is the identity matrix of size $(k+2)\times(k+2)$. This requires matrix exponentiation modulo 4, using matrices sized $(k+2)\times(k+2)$. Since we want k to be as large as possible, these matrices quickly become inconvenient, both to store, and to perform multiplications on. The step 4, presented here, instead requires mainly polynomial modulo and squaring over $\mathbb{F}_2$, and once over $\mathbb{F}_4$. This makes storage more efficient, since the size of a polynomial is a linear function of its degree. Moreover, even a naive implementation of polynomial squaring over $\mathbb{F}_2$ has time complexity $\mathcal{O}(n)$. For polynomial modulo, a naive approach has time complexity $\mathcal{O}(n^2)$, but faster algorithms are known, such as the result by Schönhage in [17], which enables polynomial division with remainder in $\mathcal{O}(n \log n \log\log n)$ time. This is better than even the fastest current matrix multiplication algorithms, such as [6], which has a complexity over $\mathcal{O}(n^{2.3})$.

4. Q-transform

A promising concept for constructing uniformly distributed high order linear recurring sequences is the application of the theory of Q-transform. In this section, we introduce some definitions and results that allow us to formulate infinite series of irreducible polynomials. Based on the idea described in Section 3, we can use such polynomials for creating uniformly distributed pseudorandom sequences with large period lengths.

During the section, q is a prime power, $\mathbb{F}$ denotes a field, $\mathbb{F}_q$ is a finite field of q elements, and $\mathbb{K}$ is an algebraic extension field of $\mathbb{F}$ or $\mathbb{F}_q$, depending on the context.

Definition 4.1. Let $p \in \mathbb{F}[x]$ be a polynomial of degree d. We say that the reciprocal polynomial of p is $p^*(x) = x^d p(x^{-1})$. We call a polynomial p self-reciprocal, if $p = p^*$.

Remark 4.2. a) If $p \in \mathbb{F}[x]$, then $p^* \in \mathbb{F}[x]$ and $(p^*)^* = p$.

b) Let $p, r, s \in \mathbb{F}[x]$ be such that $s = p \cdot r$. Then $s^* = p^* \cdot r^*$.

c) For any $p \in \mathbb{F}[x]$, p is irreducible if and only if p^* is irreducible.

d) Let $p, r, s \in \mathbb{F}[x]$ be such that $s = p \cdot r$. If p and r are self-reciprocal, then s is self-reciprocal, as well.

Proposition 1. *Let $p \in \mathbb{F}[x]$, and $\mathbb{K}$ be the splitting field of p. Then the following statements are equivalent.*

a) p is self-reciprocal;

b) $\forall \alpha \in \mathbb{K} \setminus \{0\}$: $p(\alpha) = 0$ implies $p\big(\alpha^{-1}\big) = 0$.

Corollary 4.3. *If $p \in \mathbb{F}[x]$ is self-reciprocal and irreducible of odd degree, then $p(x) = ax + a$, with some $a \in \mathbb{F}$.*

Proof. Since $\deg(p)$ is odd, Proposition 1 implies that p has a root α such that $\alpha = \alpha^{-1}$. This is possible if and only if $\alpha \in \{-1, 1\}$. Then either $x - 1$ or $x + 1$ is a divisor of p. However, p is irreducible, thus either $p = ax - a$ or $p = ax + a$, but $ax - a$ is self-reciprocal if and only if $a = -a$. □

Corollary 4.4. *Let $p \in \mathbb{F}[x]$ be a self reciprocal polynomial, and $p_1, \ldots, p_k \in \mathbb{F}[x]$ be distinct irreducible polynomials such that $p = p_1^{n_1} \cdot \cdots \cdot p_k^{n_k}$. Then for each $1 \le i \le k$ there exists $1 \le j \le k$ such that $p_i = p_j^*$ and $n_i = n_j$.*

Remark 4.5. In the previous corollary, $i = j$ if and only if p_i is self-reciprocal.

Definition 4.6. Let $p \in \mathbb{F}[x]$ be a polynomial of degree d. The Q-transform of p is $\tilde{p}(x) = x^d p(x + x^{-1})$.

Remark 4.7. If $p \in \mathbb{F}[x]$, then $\tilde{p} \in \mathbb{F}[x]$, and $\deg(\tilde{p}) = 2\deg(p)$.

Proposition 2. *Let $p, r, s \in \mathbb{F}[x]$ be such that $s = p \cdot r$. Then $\tilde{s} = \tilde{p} \cdot \tilde{r}$.*

Let $p \in \mathbb{F}[x]$, and $\alpha \in K \setminus \{0\}$. Then $\tilde{p}(\alpha) = 0$ if and only if $\tilde{p}(\alpha^{-1}) = 0$. By Proposition 1, we may state the following.

Proposition 3. *If $p \in \mathbb{F}[x]$, then $\tilde{p}$ is self-reciprocal.*

Proposition 4. *The Q-transform is an injection.*

Proof. Let $p \in \mathbb{F}[x]$, $d = \deg(p)$, $\mathbb{K}$ be the splitting field of $\tilde{p}$, and $\alpha_i, \beta_i \in \mathbb{K}$ $(i = 1, \ldots, d)$ with the following properties:

$$p(x) = a_d \prod_{i=1}^{d} (x - \alpha_i), \quad \text{and} \quad \beta_i = -\frac{1}{2}\alpha_i + \frac{1}{2}\sqrt{\alpha_i^2 - 4}\,.$$

Then

$$\tilde{p}(x) = a_d x^d \prod_{i=1}^{d} \big(x + x^{-1} - \alpha_i\big) = a_d \prod_{i=1}^{d} \big(x^2 + 1 - \alpha_i x\big)$$

$$= a_d \prod_{i=1}^{d} (x - \beta_i) \prod_{i=1}^{d} \left(x - \beta_i^{-1}\right).$$

This means that there is a one-to-one correspondence between the roots of p and the pairs of roots of $\tilde{p}$. □

Let

$$\mathcal{P}_q(d) = \{p \mid p \in \mathbb{F}_q[x], \deg(p) = d\},$$
$$\mathcal{Q}_q(d) = \{p \mid p \in \mathbb{F}_q[x], \deg(p) = 2d, p = p^*\}.$$

Since $|\mathcal{P}_q(d)| = |\mathcal{Q}_q(d)|$, Proposition 4 implies the following.

Corollary 4.8. *Let $p \in \mathbb{F}_q[x]$ be a self-reciprocal polynomial. Then there exists a unique $r \in \mathbb{F}_q[x]$ such that $p = \tilde{r}$.*

Notation 1. *Let $p \in \mathbb{F}[x]$ and $k \in \mathbb{N}$. We denote by $\tilde{p}^{(k)}$ the following iterated Q-transform:*

$$\text{if } k = 0, \text{ then } \tilde{p}^{(k)} = p;$$
$$\text{if } k > 0, \text{ then } \tilde{p}^{(k)} = \tilde{r}, \text{ where } r = \tilde{p}^{(k-1)}.$$

Corollary 4.9. *Let $p \in \mathbb{F}_q[x]$ be a self-reciprocal polynomial. Then there exists a unique $r \in \mathbb{F}_q[x]$, not a self-reciprocal polynomial, and $k \in \mathbb{N}$ such that $p = \tilde{r}^{(k)}$.*

Corollary 4.10. *Let $p \in \mathbb{F}_q[x]$ be irreducible. Then $\tilde{p}$ is either irreducible or there exist $p_1, p_2 \in \mathbb{F}_q[x]$ irreducible polynomials such that $\tilde{p} = p_1 \cdot p_2$, and $p_1 = p_2^*$.*

Proof. Assume contrary that there exists an $r \in \mathbb{F}_q[x]$ self-reciprocal polynomial with $1 \le \deg(r) < 2\deg(p)$, such that $r|\tilde{p}$. By Corollary 4.8, there exists $s \in \mathbb{F}_q[x]$ satisfying $s \mid p$, $\deg(s) < \deg(p)$, and $r = \tilde{s}$, which is a contradiction. □

Proposition 5. *Let $p \in \mathbb{F}_2[x]$ be an irreducible polynomial in the form $p(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + 1$. Then $\tilde{p}$ is irreducible if and only if $a_{d-1} = a_1 = 1$. Furthermore, the coefficient of the linear term of $\tilde{p}$ is 1.*

Proof. The proposition is proven in a more general settings in [10]. □

Corollary 4.11. *Let $p \in \mathbb{F}_2[x]$ be an irreducible polynomial, and $p(x) = x^d + x^{d-1} + a_{d-2}x^{d-2} + \cdots + a_2 x^2 + x + 1$. Then $\tilde{p}^{(k)}$ is irreducible for all $k \in \mathbb{N}$.*

This result implies that any irreducible polynomial in the form as in Proposition 5 determines an infinite sequence of irreducible Q-iterated polynomials. Every self-reciprocal polynomial of even degree is contained in exactly one of such sequences.

Proposition 6. *Let $p \in \mathbb{F}_q$ be an irreducible polynomial, accomplishing $\deg(p) = 2d$. Then p is self-reciprocal if and only if $\operatorname{ord}(p) \mid q^d + 1$.*

Proof. The proposition is stated e.g. in [5]. □

For the construction of pseudorandom number sequences with high period length, we need irreducible polynomials of high order. Actually, the period length is proportional to the order. Based on our experience, we have the following conjecture.

Conjecture 1. *Let $p \in \mathbb{F}_q[x]$ be an irreducible self-reciprocal polynomial of degree* $\deg(p) = 4d$. *Then* $q^d + 1 < \mathrm{ord}(p)$.

Furthermore, we have encountered Q-iterated polynomials having maximal order in many cases.

5. Statistical testing

In this section, we describe a test carried out to examine the statistical properties of the pseudorandom number sequences generated using the previously detailed method. Two irreducible polynomials of large degree were created, one using a brute force method and one using Q-transformations. The pseudorandom sequences generated using these polynomials were tested using the NIST statistical test suite.

The software and documentation of the NIST test suite are available at [15]. The suite includes 15 tests designed to examine the properties of pseudorandom bit sequences, such as:

- *Frequency test*: a simple check to determine the proportion of ones and zeroes in a binary sequence.
- *Runs test*: checking the number of runs (uninterrupted sequence of identical bits) of various lengths to see how closely matches the expected value in a truly random sequence.
- *DFT (Spectral) test*: determining the peak heights in the Discrete Fourier Transform of the sequence, with the purpose of finding periodic features.
- *Template matching test*: finding occurences of predetermined target strings, to detect generators producing too many such patterns. Both overlapping and non-overlapping tests are included.
- *Maurer's "Universal Statistical" test*: checking whether or not the sequence can be significantly compressed without loss of information.
- *Linear complexity test*: attempting to determine the length of the LRS that characterizes the sequence.

The first irreducible polynomial tested, denoted by t_1, was generated using irreducibility testing methods described in previous sections. The implementation uses the NTL (Number Theory Library) available at [20].

The degree of t_1 was chosen to be 216091. The reason for this choice is that $2^{216091}-1$ is a Mersenne prime. Choosing a value this way simplifies Step 4 of the algorithm described in Section 2. Note that this step requires the computation of the order of the irreducible polynomial, which is a divisor of 2^d-1, where d is the degree of the polynomial. If d is large, this step becomes computationally impractical, but choosing 2^d-1 to be a prime gives a simple solution to the problem.

The second irreducible polynomial tested, denoted t_2, was created using iterated Q-transform, using the following method:

1. Let q be a self-reciprocal irreducible monic polynomial, with $\deg(q)=d$.
2. Run the algorithm described in Section 3, using q as input. Let p be the candidate polynomial that remains after Step 4. Determine $s,r\in\mathbb{Z}[x]$ such that $p=sq+r$, and $\deg(r)<\deg(q)$.
3. Compute $t=s\tilde{q}^{(n)}+r$, where $\tilde{q}^{(n)}$ is the iterated Q-transform, described in Notation 1 of Section 4. Use t to construct the linear recurrence sequence.

Based on practical observation, if the sequence produced by p has uniform distribution, then the sequence produced by t will also have uniform distribution. However, the proof of this conjecture is currently an open question.

To create t_2, the following polynomial was used as a starting point:

$$q_2 = x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9 \\ +x^7+x^5+x^4+x^3+x^2+x+1\,.$$

As it is stated in Proposition 6, the order of a self-reciprocal irreducible monic polynomial of degree d is at most $2^{\frac{d}{2}}+1$. The above polynomial was chosen because $\mathrm{ord}(q),\mathrm{ord}(\tilde{q})$, and $\mathrm{ord}(\tilde{q}^{(2)})$ all reach this maximum value. Theoretically, it does not guarantee that this maximality property will hold after further Q-transformations, but practical observations suggest that the order of $\tilde{q}^{(n)}$ will grow at a rate that is sufficient for use in the applications described in this paper. We stated our related experience in Conjecture 1.

Using the above method, $p_2=s_2q_2+r_2$ was determined, and the polynomial to be used was set as $t_2=s_2\tilde{q_2}^{(14)}+r_2$. Note that $\deg(\tilde{q_2}^{(14)})=229376$, and $\deg(t_2)=229378$.

Using t_1 and t_2, two LRSs were created to generate the pseudorandom sequences to be tested, denoted L_1 and L_2 respectively. Both LRSs generate 64-bit words. Following the recommendations in the documentation of the NIST test suite, 16MB (2^{21} words) of test data were generated using L_1 and L_2 each.

For each of these two streams, the NIST suite split the data into 100 bitstreams. The testing software provides a detailed output of the tests, as well as a summary showing the number of bitstreams that passed each test. The minimum pass rate for a test is considered to be 96 out of a sample size of 100.

Tables 1 and 2 show some of the result obtained from the tests. The full report can be found at https://arato.inf.unideb.hu/major.sandor/statistical_results/.

Table 1. NIST test results of L_1 generator.

Statistical Test	P-value	Proportion
Frequency	0.779188	100/100
Runs	0.514124	100/100
FFT	0.924076	99/100
OverlappingTemplate	0.012650	96/100
Universal	0.935716	97/100
LinearComplexity	0.699313	99/100

Table 2. NIST test results of L_2 generator.

Statistical Test	P-value	Proportion
Frequency	0.955835	100/100
Runs	0.108791	98/100
FFT	0.678686	98/100
OverlappingTemplate	0.035174	97/100
Universal	0.249284	100/100
LinearComplexity	0.719747	100/100

The results show the two generators to have very similar statistical properties, with L_2 being only slightly weaker in some tests. Since the order of t_2 is significantly lower than the order of t_1, this result is to be expected. Also of note is that both generators produced very high quality pseudorandom sequences, passing all relevant benchmarks set by the test suite.

This shows that using the Q-transformation described above to generate irreducible polynomials of very large degree is completely suitable for use in generating uniformly distributed pseudorandom linear recurrence sequences.

References

[1] M. Ben-Or: *Probabilistic Algorithms in Finite Fields*, 22nd Annual Symposium on Foundations of Computer Science (1981), pp. 394–398.

[2] E. R. Berlekamp: *Factoring Polynomials over Finite Fields*, The Bell System Technical Journal 46(8) (1967), pp. 1853–1859.

[3] D. Cantor, H. Zassenhaus: *A New Algorithm for Factoring Polynomials Over Finite Fields*, Mathematics of Computation 36(154) (1981), pp. 587–592.

[4] B. Chor, R. L. Rivest: *A Knapsack-type Public Key Cryptosystem Based on Arithmetic in Finite Fields*, IEEE Transactions on Information Theory 34(5) (1988), pp. 901–909.

[5] S. Cohen: *Polynomials over finite fields with large order and level*, Bull. Korean Math. Soc. 24(2) (1987), pp. 83–96.

[6] F. L. Gall: *Powers of tensors and fast matrix multiplication*, Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC '14) (2014), pp. 296–303.

[7] S. Gao, D. Panario: *Tests and Constructions of Irreducible Polynomials over Finite Fields*, Foundations of Computational Mathematics (1997).

[8] T. Herendi: *Construction of Uniformly Distributed Linear Recurring Sequences Modulo Power of 2*, Uniform Distribution Theory 13(1) (2018), pp. 109–129.

[9] R. Lidl, H. Niederreiter: *Introduction to Finite Fields and their Applications*, Cambridge: Cambridge University Press, 1994.

[10] H. Mein: *On the Construction of Irreducible Self-Reciprocal Polynomials Over Finite Fields*, Communication and Computing 1 (1990), pp. 43–53.

[11] A. J. Menezes, S. A. Vanstone, P. C. V. Oorschot: *Handbook of Applied Cryptography (1st. ed.)* USA: CRC Press, Inc., 1996.

[12] S. L. Miller, D. Childers: *Probability and Random Processes (2nd. ed.)* Boston: Academic Press, 2012, pp. 517–546.

[13] P. Naudin, Q. Claude: *Univariate polynomial factorization over finite fields*, Theoretical Computer Science 191 (1998), pp. 1–36.

[14] H. Niederreiter, Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics, 1992, pp. 161–176.

[15] NIST: *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, URL: https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software.

[16] M. Rabin: *Probabilistic Algorithms in Finite Fields*, SIAM J. Comput. 9 (1980), pp. 273–280.

[17] A. Schönhage: *Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients* (1982).

[18] V. Shoup: *Fast Construction of Irreducible Polynomials over Finite Fields*, Journal of Symbolic Computation 17(5) (1994), pp. 371–391.

[19] V. Shoup: *New Algorithm for Finding Irreducible Polynomials over Finite Fields*, Mathematics of Computation 54(189) (1990), pp. 435–447.

[20] V. Shoup: *NTL: A Library for doing Number Theory*, URL: https://libntl.org/.