Mathematical Problems of Computer Science 36, 57-62, 2012.

Computation of the Complexity of some Recursive Constructed Normal Polynomials

Mahmood Alizadeh

Islamic Azad University- Ahvaz Branch
E-mail: Alizadeh@iauahvaz.ac.ir

Abstract

In this paper we give some algorithms for computing the complexity of some normal polynomials constructed by some recurrent methods. Finally some results of our algorithms are given in a table.

Keywords: Complexity, Irreducible Polynomial, Normal Polynomial.

1. Introduction

For a prime power $q = p^s$ and a positive integer n, let F_q and F_{q^n} be finite fields. A normal basis N for F_{q^n} over F_q is a basis of the form $N = \{\alpha, \alpha^q, \alpha^{q^2}, ..., \alpha^{q^{n-1}}\}$ for some element $\alpha \in F_{q^n}$. the element $\alpha \in N$ is called a normal element of F_{q^n} over F_q. A monic irreducible polynomial $f(x) \in F_q[x]$ of degree n is called a normal polynomial (N-polynomial) if it is the minimal polynomial of some normal element. As the elements in a normal basis are exactly the roots of some N-polynomial, there is a canonical one-to-one correspondence between N-polynomials and normal basis.

one problem in general is: given an integer n and the ground field F_q, construct a normal basis of F_{q^n} over F_q, or, equivalently, construct an N-polynomial in $F_q[x]$ of degree n.

Some results regarding computationally simple constructions of N-polynomials over F_q can be found in [3, 4, 5, 7, 8]. With the development of coding theory and the appearance of several cryptosystems using finite fields, the implementation of finite field arithmetic, in either hardware or software, designs or implementations, including single-ship exponentiators for the fields $F_{2^{127}}$, $F_{2^{155}}$, $F_{2^{332}}$, and an encryption processor for $F_{2^{593}}$ for public key cryptography. These products are based on multiplication schemes due to Massey and Omura [10] and Mullin, Onyszchuk and Vanstone [11] by using normal basis to represent finite fields and choosing appropriate algorithms for the arithmetic of course. the advantages of using a normal basis representation has been known for many years. The complexity of the hardware design of such multiplication schemes is heavily dependent on the choice of the normal bases used.

In this paper we give some algorithms for computing the complexity of some normal basis or equivalently some normal polynomials constructed by some recurrent methods. Finally some results in a table are given.

2. Preliminaries

We state now some results that will be helpful to derive our results.

Theorem 1 *(M.K.Kyuregyan [5],Theorem) For* $q = 2^s$, *let* $F_0(x) = \sum_{i=0}^{n} c_i x^i$ *be a N-polynomial of degree n over* F_q *whose coefficients satisfy the conditions* $\sum_{v=0}^{s-1} (\frac{c_1}{c_0})^{2^v} = 1$ *and* $\sum_{v=0}^{s-1} c_{n-1}^{2^v} = 1$. *Then, the sequence* $(F_k(x))_{k \geq 0}$ *defined by*

$$F_{k+1} = x^{n2^k} F_k(x + x^{-1}), \quad k \geq 0$$

is a sequence of N-polynomials of degree $n2^k$ *over* F_{2^s}.

Theorem 2 *([1],Theorem 2) Let* $P(x) = \sum_{i=0}^{n} c_i x^i$ *be an irreducible polynomial of degree n over* F_{2^s} *and* $P^*(x)$ *be a N-polynomial over* F_{2^s}. *Also let*

$$F(x) = (x^2 - x + 1)^n P(\frac{x^2 - x}{x^2 - x + 1}). \tag{1}$$

Then $F^*(x)$ *is an N-polynomial of degree 2n over* F_{2^s} *if and only if*

$$(n + \frac{c_1}{c_0}) \cdot Tr_{2^s|2}(\frac{P'(1)}{P(1)} - n) \neq 0.$$

Let us now look at how the addition and multiplication in F_{q^n} can be done in general. We view F_{q^n} as a vector space of dimension n over F_q. Let $\alpha_1, \alpha_2, ... \alpha_{n-1} \in F_{q^n}$ be linearly independent over F_q. Then every element $A \in F_{q^n}$ can be represented as $A = \sum_{i=0}^{n-1} a_i \alpha_i$, $a_i \in F_q$. Thus F_{q^n} can be identified as $F_q{}^n$, the set of all n-tuple over F_{q^n}, and $A \in F_{q^n}$ can written as $A = (a_0, a_1, ..., a_{n-1})$. Let $B = (b_0, b_1, ..., b_{n-1})$ be another element in F_{q^n}. Then addition is component-wise and is easy to implement. Multiplication is more complicated. Let $A \cdot B = C = (c_0, c_1, ..., c_{n-1})$. We wish to express the c_i's as simply as possible in terms of the a_i's and b_i's. Suppose

$$\alpha_i \cdot \alpha_j = \sum_{k=0}^{n-1} t_{ij}^{(k)} \alpha_k \quad t_{ij}^{(k)} \in F_q. \tag{2}$$

Then it is easy to see that

$$c_k = \sum_{i,j} a_i b_j t_{ij}^{(k)} = A T_k B^t, \quad 0 \leq k \leq n-1,$$

where $T_k = (t_{ij}^{(k)})$ is an $n \times n$ matrix over F_q and B^t is the transpose of B. The collection of matrices $\{T_k\}$ is called a multiplication table for F_{q^n} over F_q. Observe that the matrices $\{T_k\}$ are independent of A and B. In the following we examine the Massey Omura scheme which exploits the symmetry of normal bases.
Let $N = \{\alpha_0, \alpha_1, ..., \alpha_{n-1}\}$ be a normal basis of F_{q^n} over F_q where $\alpha_i = \alpha^{q^i}$. Then $\alpha_i^{q^k} = \alpha_{i+k}$ for any integer k, where indices of α are reduced module n. Let us first consider the operation of exponentiation by q. The element A^q has coordinate vector $(a_{n-1}, a_0, a_1, ..., a_{n-2})$. That is, the coordinates of A^q are just a cyclic shift of the coordinates of A, and so the cost of computing A^q is negligible.

Let $t_{ij}^{(k)}$ terms be defined by (2). Raising both side of equation (2) to the $q^{-l}-th$ power, one finds that

$$t_{ij}^{(l)} = t_{i-l,j-l}^{(0)}, \quad 0 \leq i,j,l \leq n-1.$$

consequently, if a circuit is built to compute c_0 with input A and B, then the same circuit with input $A^{q^{-l}}$ and $B^{q^{-l}}$ yields the product terms c_l ($A^{q^{-l}}$ and $B^{q^{-l}}$ are simply cyclic shifts of the vector representations of A and B). Thus each term of C is successively generated by shifting the A and B vectors, and thus C is calculated in n clock cycles. The number of gates required in this circuit equals the number of nonzero entries in the matrix T_0. Let

$$\alpha \cdot \alpha_i = \sum_{j=0}^{n-1} t_{ij}\alpha_j \quad 0 \leq i \leq n-1, \; t_{ij} \in F_q. \tag{3}$$

Let the $n \times n$ matrix (t_{ij}) be denoted T. It is easy to prove that

$$t_{ij}^{(k)} = t_{i-j,k-j}, \quad for\, all \;\; i,j,k.$$

Therefore the number of non-zero entries in T_0 is equal to the number of non-zero entries in T. Following Mullin, Onyszchuk, Vanstone and Wilson [9], we call the number of non-zero entries in T the complexity of the normal basis N (or the complexity of the normal polynomial $f(x)$ corresponding to N) and denote it by C_N.

3. Algorithms and Results

The following algorithm, that computes the complexity of given a normal polynomial is given in [2].

Algorithm 1([2], Algorithm 2.2)
Input: Given an N-polynomial $f(x)$ of degree n over F_q
Out put: The Complexity C_N of the N-polynomial $f(x)$

.Set $C_N = 0$
.Set $r_1(x) = x$
.Set $k_1(x) \equiv x \cdot r_1(x) \pmod{f(x)}$
  .For $i = 2:n$
  -Set $r_i(x) \equiv (r_{i-1}(x))^q \pmod{f(x)}$
  -Set $k_i(x) \equiv x \cdot r_i(x) \pmod{f(x)}$
  .End for
  .For $i = 1:n$
  -Find solution $T_i = (t_{i1}, t_{i2}, \ldots, t_{in})$ of the linear equation system $k_i(x) = \sum_{j=1}^{n} t_{ij} r_j(x)$
    -For $j = 1:n$
      -If $t_{ij} \neq 0$
      -Set $C_N = C_N + 1$
      -End if
    -End for
  .End for
.Return C_N.

Remark Let $f(x)$ be a N-polynomial of degree n over F_q and α be a root of $f(x)$ in F_{q^n}. So $N = \{\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}\}$ is a normal basis for F_{q^n} over F_q. By (3), we know that the complexity of $f(x)$ is the number of nonzero elements t_{ij}'s such that

$$\alpha^{q^{i-1}+1} = \sum_{j=1}^{n} t_{ij}\alpha^{q^{j-1}} \quad 1 \leq i \leq n, \; t_{ij} \in F_q. \tag{4}$$

Obviously for computing complexity of the normal polynomial $f(x)$, the only point that remains to be settled is how to compute t_{ij}'s in (4). This can be achieved as follows: For $1 \leq i \leq n$, calculate the unique polynomials $r_i(x)$ and $k_i(x)$ of degree less than n with $x^{q^{i-1}} \equiv r_i(x) \bmod f(x)$ and $x \cdot r_i(x) \equiv k_i(x) \bmod f(x)$. (We note that for computing t_{ij}'s we should calculate $\alpha^{q^{i-1}}$ and $\alpha^{q^{i-1}+1}$, but $x^{q^{i-1}}$ and $x^{q^{i-1}+1}$ can be very high degree polynomials. Reducing module $f(x)$ just keeps the degree reasonable).
Then determine elements $t_{ij} \in F_q$, such that

$$k_i(x) = \sum_{j=1}^{n} t_{ij} r_j(x) \quad 1 \leq i \leq n. \tag{5}$$

This involves n conditions concerning the vanishing of the coefficients of x^j, $1 \leq j \leq n$, and thus leads to a homogeneous system of n linear equations. So lines 2 until 7 of Algorithm 1 compute $\alpha^{q^{i-1}+1}$ and $\alpha^{q^{i-1}}$ module $f(x)$ (or $k_i(x)$ and $r_j(x)$ respectively) for $1 \leq i, j \leq n$, that are necessary for computing (5). Also lines 8 until 15 compute the number of nonzero elements t_{ij}'s, satisfied in (5).

Example Consider the N-polynomial $f(x) = x^5 + x^4 + x^2 + x + 1$ over F_2. We have $r_1(x) = x, r_2(x) = x^2, r_3(x) = x^4, r_4(x) = x^3 + x, r_5(x) = x^4 + x^3 + x^2 + 1$ and $k_1(x) = x^2, k_2(x) = x^3, k_3(x) = x^4 + x^2 + x + 1, k_4(x) = x^4 + x^2, k_5(x) = x^3 + x^2 + 1$. So by (5) we have

$$t_{11}x + t_{12}x^2 + t_{13}x^4 + t_{14}(x^3 + x) + t_{15}(x^4 + x^3 + x^2 + 1) = x^2$$

or

$$t_{15} + (t_{11} + t_{14})x + (t_{12} + t_{15})x^2 + (t_{14} + t_{15})x^3 + (t_{13} + t_{15})x^4 = x^2.$$

Therefore we have the following equation system:

$$\begin{aligned} t_{15} &= 0 \\ t_{11} + t_{14} &= 0 \\ t_{12} + t_{15} &= 1 \\ t_{14} + t_{15} &= 0 \\ t_{13} + t_{15} &= 0 \end{aligned}$$

So we have $t_{11} = t_{13} = t_{14} = t_{15} = 0$ and $t_{12} = 1$.
Similarly we compute all of t_{ij}'s, and so we have $t_{12} = t_{21} = t_{24} = t_{34} = t_{35} = t_{42} = t_{43} = t_{53} = t_{55} = 1$ and the others of t_{ij}'s are equal to zero. Hence the complexity of $f(x)$ is $C_N = 9$.

The following algorithms compute the complexity of normal polynomials $F_k(x)$ constructed by Theorems 1 and 2 for every $k \geq 0$.

Algorithm 2

Input: Given an N-polynomial $P(x) = \sum_{i=0}^{n} c_i x^i$ of degree n over F_{2^s}, integer k.

Out put: Complexity C_N of normal polynomial $F_k(x)$ constructed by Theorem 1.

1) .Set $F_0(x) = P(x)$.
2) .If $\sum_{i=0}^{s-1} (\frac{c_1}{c_0})^{2^i} = 1$ and $\sum_{i=0}^{s-1} c_{n-1}^{2^i} = 1$
3) -For $m = 0 : k-1$
4) -Set $F_{m+1} = x^{n2^m} F_m(x + x^{-1})$.
5) -End for
6) -Set Algorithm 1 for $f(x) = F_k(x)$ and $p = 2$.
7) .Else if
8) -Print(Theorem's hypothesis is not satisfied)
9) .End if
10) .Return C_N.

Algorithm 3

Input: Given an N-polynomial $P(x) = \sum_{i=0}^{n} c_i x^i$ of degree n over F_{2^s}, integer k.

Out put: The complexity of normal polynomials $F_k(x)$ constructed by Theorem 2 .

1) .Set $F_0(x) = P^*(x)$.
2) .If $Tr_{2^s|2}(\frac{P^{*\prime}(1)}{P^*(1)} - n) \cdot Tr_{2^s|2}(\frac{c_{n-1}}{c_n} - n) \neq 0$.
3) -For $m = 0 : k-1$
4) -Set $F_{m+1} = (x^2+x+1)^{n2^m} F_m(\frac{x^2+x}{x^2+x+1})$.
5) -End for
6) -Set $G_k(x) = x^{n2^k} F_k(\frac{1}{x})$.
7) -Set Algorithm 1 for $f(x) = G_k(x)$ and $p = 2$.
8) .Else if
9) -Print(Conditions of theorem is not satisfied)
10) .End if
11) .Return C_N.

Some results of the above Algorithms in the following table is given.

Table 1: The complexity C_N of normal polynomials $F_k(x)$ constructed by Theorems 1 and 2 for $k \leq 8$, with $F_0(x) = x^2 + x + 1$ over F_2.

k	1	2	3	4	5	6	7	8
$deg(F_k(x))$	4	8	16	32	64	128	256	512
C_N(For $F_k(x)$ constructed by Thm1)	7	27	115	479	2011	8247	32407	131155
C_N(For $F_k(x)$ constructed by Thm2)	9	25	97	453	1921	7941	32229	130465

References

[1] S. Abrahamyan, "Some construction of N-polynomials over finite fields". *National Academy of Sciences of Armenia Reports*, vol. 111, no. 3, 232-239. 2011.

[2] M. Alizadeh, "Some algorithms for normality testing irreducible polynomials and computing complexity of the normal polynomials over finite fields. *Applied Mathematical Sciences*. vol. 6, no. 40. 1997 - 2003. 2012.

[3] S. Gao."Normal bases over finite fields", Ph.D. Thesis, Waterloo, 1993.

[4] M. K. Kyuregyan. "Iterated construction of irreducible polynomials over finite fields with linearly independent roots",*Finite Fields Appl.* , vol. 10, pp. 323-341, 2004.

[5] M. K. Kyuregyan,"Recursive constructions of N-polynomials over $GF(2^s)$", *Discrete Applied Mathematics*, vol. 156, pp. 1554-1559, 2008.

[6] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, 1987.

[7] A. J. Menezes, I. F.Blake , X.Gao, R. C. Mullin, S. A. Vanstone and T. Yaghoobian, *Applications of finite fields*, Kluwer Academic publishers , Boston , Dordrecht, Lancaster , 1993.

[8] H. Meyn, "Explicit N-polynomial of 2-power degree over finite fields", *Designs, Codes and Cryptography*, vol. 6, pp. 147-158, 1995.

[9] R. Mullin, I. Onyszchuk, S. Vanstone and R. Wilson,"Optimal normal basesin $GF(p^n)$"",*Discreate Applied math.*,vol. 22, ,149-161, 1988/1989.

[10] J. Omura and J. Massey, "Computational method and apparatus for finite field arithmatic",U.S. patent 4,586,627, 1986.

[11] I. Onyszchuk, R. Mullin, and S. Vanstone, "Computational method and apparatus for multiplication",U.S patent 4,745,568, 1988.

Ռեկուրսիվ կառուցված որոշ նորմալ բազմանդամների բարդության հաշվումը

Մ. Ալիզադեհ

Ամփոփում

Այս աշխատանքում մենք առաջարկում ենք որոշակի ալգորիթմ հաշվելու համար որոշ նորմալ բազմանդամների բարդությունը, որոնք կառուցվել են ռեկուրենտ եղանակներով: Բերված է աղյուսակ, որում նշված է որոշ արդյունքներ:

Вычисление сложности некоторых рекурсивно построенных полиномов

М. Ализаде

Аннотация

В этой статье мы предлагаем алгоритмы вычисления сложности нормальных полиномов, построенных определенными рекуррентными методами.

В заключении приведена таблица, которая содержит некоторые результаты.