Efficient multiplication using type 2 optimal normal bases

Joachim von zur Gathen[1], Amin Shokrollahi[2], and Jamshid Shokrollahi[3]*

[1] B-IT, Dahlmannstr. 2, Universität Bonn, 53113 Bonn, Germany
gathen@bit.uni-bonn.de
[2] ALGO, Station 14, Batiment BC, EPFL, 1015 Lausanne, Switzerland
amin.shokrollahi@epfl.ch
[3] B-IT, Dahlmannstr. 2, Universität Bonn, 53113 Bonn, Germany
jamshid@bit.uni-bonn.de

Abstract. In this paper we propose a new structure for multiplication using optimal normal bases of type 2. The multiplier uses an efficient linear transformation to convert the normal basis representations of elements of $\mathbb{F}_{q^n}$ to suitable polynomials of degree at most n over $\mathbb{F}_q$. These polynomials are multiplied using any method which is suitable for the implementation platform, then the product is converted back to the normal basis using the inverse of the above transformation. The efficiency of the transformation arises from a special factorization of its matrix into sparse matrices. This factorization — which resembles the FFT factorization of the DFT matrix — allows to compute the transformation and its inverse using $O(n \log n)$ operations in $\mathbb{F}_q$, rather than $O(n^2)$ operations needed for a general change of basis. Using this technique we can reduce the asymptotic cost of multiplication in optimal normal bases of type 2 from $2\mathsf{M}(n) + O(n)$ reported by Gao et al. (2000) to $\mathsf{M}(n) + O(n \log n)$ operations in $\mathbb{F}_q$, where $\mathsf{M}(n)$ is the number of $\mathbb{F}_q$-operations to multiply two polynomials of degree $n-1$ over $\mathbb{F}_q$. We show that this cost is also smaller than other proposed multipliers for $n > 160$, values which are used in elliptic curve cryptography.

Keywords: Finite field arithmetic, optimal normal bases, asymptotically fast algorithms.

1 Introduction

The normal basis representation of finite fields enables easy computation of the qth power of elements. Assuming q to be a prime power, a basis of the form $(\alpha, \alpha^q, \cdots, \alpha^{q^{n-1}})$ for $\mathbb{F}_{q^n}$, as a vector space over $\mathbb{F}_q$, is called a normal basis generated by the normal element $\alpha \in \mathbb{F}_{q^n}$. In this basis the qth power of an element can be computed by means of a single cyclic shift. This property makes such bases attractive for parallel exponentiation in finite fields (see Nöcker 2001).

* Partially funded by the German Research Foundation (DFG) under project RU 477/8

Naive multiplication in these bases is more expensive than in polynomial bases, especially when using linear algebra (cf. Mullin et al. (1989)). Hence substantial effort has gone into reducing the multiplication cost. In this paper a new method for multiplication in normal bases of type 2 is suggested. It uses an area efficient circuit to convert the normal basis representation to polynomials and vice versa. Any method can be used to multiply the resulting polynomials. Although this structure has small area, its propagation delay is longer than other methods and, when used in cryptography, is mostly suitable for applications where the area is limited, like in RFIDs.

One popular normal basis multiplier is the Massey-Omura multiplier presented by Omura & Massey. The space and time costs of this multiplier increase with the number of nonzero coefficients in the matrix representation of the endomorphism $x \rightarrow \alpha x$ over $\mathbb{F}_{q^n}$, where α generates the normal basis. Mullin et al. (1989) show that this number is at least $2n - 1$ which can be achieved for optimal normal bases. Gao & Lenstra (1992) specify exactly the finite fields for which optimal normal bases exist. They are related to Gauss periods, and can be grouped into optimal normal bases of type 1 and 2.

For security reasons only prime extension degrees are used in cryptography, whereas the extension degrees of the finite fields containing an optimal normal basis of type 1 are always composite numbers. Cryptography standards often suggest finite fields for which the type of normal bases are small (see for example the Digital Signature Standard (2000)) to enable designers to deploy normal bases. Applications in cryptography have stimulated research about efficient multiplication using optimal normal bases of type 2. The best space complexity results for the type 2 multipliers are n^2 and $3n(n-1)/2$ gates of types AND and XOR, respectively reported by Sunar & Koç (2001) and Reyhani-Masoleh & Hasan (2002). Their suggested circuits are obtained by suitably modifying the Massey-Omura multiplier. A classical polynomial basis multiplier, however, requires only n^2 and $(n-1)^2$ gates of types AND and XOR respectively for the polynomial multiplication, followed by a modular reduction. The latter is done using a small circuit of size of $(r-1)n$, where r is the number of nonzero coefficients in the polynomial which is used to create the polynomial basis. It is conjectured by von zur Gathen & Nöcker (2004) that there are usually irreducible trinomials or pentanomials of degree n. The above costs and the fact that there are asymptotically fast methods for polynomial arithmetic suggest the use of polynomial multipliers for normal bases to make good use of both representations. The proposed multiplier in this paper works for normal bases but its space complexity is similar to that of polynomial multipliers. Using classical polynomial multiplication methods, it requires $2n^2 + 16n \log_2 n$ gates in $\mathbb{F}_{2^n}$. With the Karatsuba algorithm, we can decrease the space asymptotically even further down to $O(n^{\log_2 3})$. The usefulness of this approach in hardware has first been demonstrated in Grabbe et al. (2003). The proposed structure can be employed to compute of Tate-pairing in characteristic three, for example. Applications of optimal normal bases of type 2 for pairing-based cryptography have been proposed by Granger et al. (2005).

The connection between polynomial and normal bases, together with its application in achieving high performance multiplication in normal bases, has been investigated by Gao et al. (1995, 2000). The present work can be viewed as a conceptual continuation of the approach in those papers. They describe how multiplication in normal basis representation by Gauss periods for $\mathbb{F}_{q^n}$ can be reduced to multiplication of two $2n$-coefficient polynomials, which because of the existing symmetries can be done by two multiplications of n-coefficient polynomials: here any method, including asymptotically fast ones, can be deployed.

The multiplier of this work is based on a similar approach. For optimal normal bases of type 2 we present an efficient transformation which changes the representations from the normal basis to suitable polynomials. These polynomials are multiplied using any method, such as the classical or the Karatsuba multiplier. Using the inverse transformation circuit and an additional small circuit the result is converted back into the normal basis representation. The heart of this method is a factorization of the transformation matrix between the two representations into a small product of sparse matrices. The circuit requires roughly $O(n \log n)$ operations in $\mathbb{F}_q$ and resembles the circuit used for computing the Fast Fourier Transformation (FFT). The analogy to the FFT circuit goes even further: as with the FFT, the inverse of the transformation has a very similar circuit. It should be noted that a general basis conversion requires $O(n^2)$ operations, as also reported by Kaliski & Liskov (1999). Recently Fan & Hasan (2006) found a new multiplier for normal bases with asymptotically low cost of $O(n^{\log_2 3})$, which uses fast multiplication methods by Toeplitz matrices. One advantage of our multiplier is the ability of working with any polynomial multiplication. Hence using the Cantor multiplier, we can achieve a cost of $O(n(\log n)^2(\log \log n)^3)$.

This paper begins with a review of Gauss periods and normal bases of type 2. Then the structure of the multiplier is introduced and the costs of each part of the multiplier are computed. The last section compares the results with the literature.

In this Extended Abstract most of the proofs and also some possible improvements for fields of characteristic 2 are omitted for lack of space. The only exception is Lemma 1 which is a central part of this paper. But we have tried to intuitively describe why the theorems are correct. The proofs can be found in the full paper, or the work of Shokrollahi (2006).

2 Permuted normal basis

It is well known (see Wassermann (1990), Gao et al. (2000), Sunar & Koç (2001)) that a type 2 optimal normal basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ is of the form

$$\mathcal{N} = (\beta + \beta^{-1}, \beta^q + \beta^{-q}, \cdots, \beta^{q^{n-1}} + \beta^{-q^{n-1}}), \qquad (1)$$

where β is a $2n+1$st primitive root of unity in $\mathbb{F}_{q^{2n}}$, and that the basis

$$\mathcal{N}' = (\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \cdots, \beta^n + \beta^{-n}),$$

which we call the permuted normal basis, is a permutation of $\mathcal{N}$. Hence the normal basis representation of an element $a = \sum_{k=0}^{n-1} a_k^{(\mathcal{N})}(\beta^{q^k} + \beta^{-q^k}) \in \mathbb{F}_{q^n}$ can be written as

$$a = \sum_{l=1}^{n} a_l^{(\mathcal{N}')}(\beta^l + \beta^{-l}), \tag{2}$$

where $(a_l^{(\mathcal{N}')})_{1\leq l\leq n}$ is a permutation of $(a_k^{(\mathcal{N})})_{0\leq k<n}$, called the permuted normal representation of a. The $a_k^{(\mathcal{N})}$ and $a_l^{(\mathcal{N}')}$ are elements of $\mathbb{F}_q$.

3 Multiplier Structure

The structure of the multiplier is described in Figure 1. To multiply two elements $a, b \in \mathbb{F}_{q^n}$ given in the basis (1) we first convert their representations to the permuted form as

$$a = \sum_{i=1}^{n} a_i^{(\mathcal{N}')}(\beta^i + \beta^{-i}), \text{ and } b = \sum_{i=1}^{n} b_i^{(\mathcal{N}')}(\beta^i + \beta^{-i}).$$

By inserting a zero at the beginning of the representation vectors and applying a linear mapping π_{n+1}, which we define in Section 4, from $\mathbb{F}_q^{n+1}$ to $\mathbb{F}_q[x]^{\leq n}$, the vectors of these representations are converted to polynomials ϕ_a and ϕ_b of degree at most n, such that their values at $\beta + \beta^{-1}$ are a and b, respectively. Then ϕ_a and ϕ_b are multiplied using an appropriate method with respect to the polynomial degrees and implementation platform. Obviously, the value of the resulting polynomial ϕ_c at $\beta+\beta^{-1}$ is the product $c = a\cdot b$. The degree of ϕ_c is at most $2n$, and the evaluation is a linear combination of $(\beta+\beta^{-1})^i$ for $0 \leq i \leq 2n$. Using another linear mapping ν_{2n+1} from $\mathbb{F}_q[x]^{\leq 2n}$ to $\mathbb{F}_q^{2n+1}$, namely the inverse of π_{2n+1}, ϕ_c is converted to a linear combination of the vectors 1 and $\beta^i+\beta^{-i}$ for $1 \leq i \leq 2n$. This is then converted to the permuted normal basis using another linear mapping τ_n.

The linear mapping ν_{2n+1} takes a polynomial in $\mathbb{F}_q[x]^{\leq 2n}$, evaluates it at $\beta + \beta^{-1}$, and represents the result as a linear combination of 1 and $\beta^i + \beta^{-i}$, for $1 \leq i \leq 2n$. Since the above vectors are linearly dependent there are several choices for ν_{2n+1}. One way to compute the resulting linear combination is to expand $(\beta + \beta^{-1})^j$, for $1 \leq j \leq 2n$, as a linear combination of $\beta^i + \beta^{-i}$, for $1 \leq i \leq 2n$. The coefficients of these expansions are closely connected to the binomial coefficients, that is, the entries of the Pascal triangle. The matrix representation of ν_{2n+1} has a structure similar to the Pascal triangle reduced modulo p, the characteristic of $\mathbb{F}_q$. This infinite triangle has a fractal structure, which has attracted a lot of attention and has been given various names in the literature, among them "Sierpinski triangle" or "Sierpinski gasket" (see Wikipedia) for $p = 2$. The central result of this paper is in Section 5, where we find a special factorization for the matrix representation of ν_{2n+1} in an appropriate basis which allows the mapping to be computed in $O(n \log n)$ operations. This cost is also sufficient for π_{n+1}.

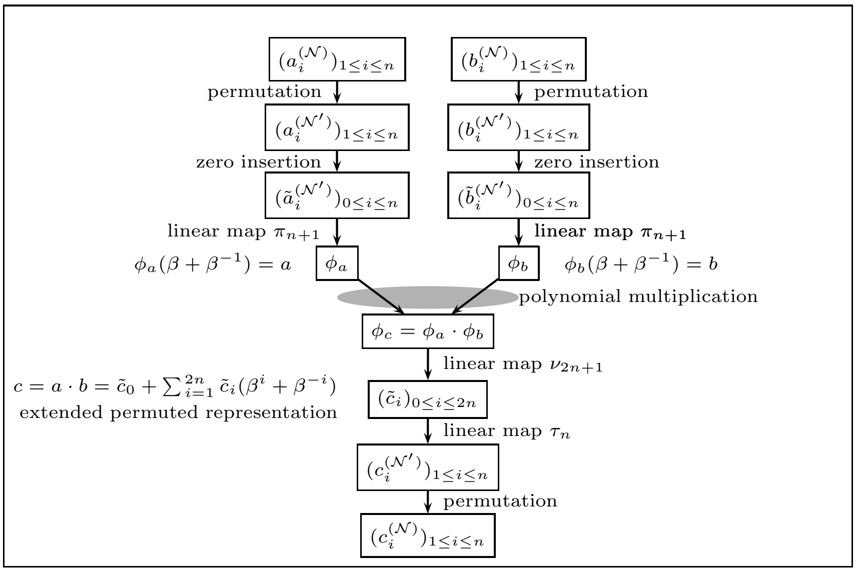

$(a_i^{(\mathcal{N})})_{1\leq i\leq n}$
$(b_i^{(\mathcal{N})})_{1\leq i\leq n}$
permutation
permutation
$(a_i^{(\mathcal{N}')})_{1\leq i\leq n}$
$(b_i^{(\mathcal{N}')})_{1\leq i\leq n}$
zero insertion
zero insertion
$(\tilde{a}_i^{(\mathcal{N}')})_{0\leq i\leq n}$
$(\tilde{b}_i^{(\mathcal{N}')})_{0\leq i\leq n}$
linear map π_{n+1}
linear map π_{n+1}
$\phi_a(\beta+\beta^{-1}) = a$
ϕ_a
ϕ_b
$\phi_b(\beta+\beta^{-1}) = b$
polynomial multiplication
$\phi_c = \phi_a \cdot \phi_b$
linear map ν_{2n+1}
$c = a \cdot b = \tilde{c}_0 + \sum_{i=1}^{2n} \tilde{c}_i(\beta^i + \beta^{-i})$
extended permuted representation
$(\tilde{c}_i)_{0\leq i\leq 2n}$
linear map τ_n
$(c_i^{(\mathcal{N}')})_{1\leq i\leq n}$
permutation
$(c_i^{(\mathcal{N})})_{1\leq i\leq n}$

Fig. 1. Overview of our multiplier structure to multiply two elements $a, b \in \mathbb{F}_{q^n}$ in the representation $(*_i^{(\mathcal{N})})_{1\leq i\leq n}$ with respect to the normal basis $\mathcal{N}$

4 Polynomials from Normal Bases

In this paper we always represent the characteristic of $\mathbb{F}_q$ by p.

The most important parts of our multiplier are the converters between polynomial and permuted normal representations. Since the elements $(\beta+\beta^{-1})^i$, for $0 \leq i \leq n$, and also 1 and $\beta^i+\beta^{-i}$, for $1 \leq i \leq 2n$, are linearly dependent, there are different possibilities for the maps π_{n+1} and ν_{2n+1} from Section 3. We define our selection via matrices $P_{n+1} \in \mathbb{F}_p^{(n+1)\times(n+1)}$ and $L_{2n+1} \in \mathbb{F}_p^{(2n+1)\times(2n+1)}$. These matrices have special factorizations which allow to multiply them by vectors of appropriate length using $O(n \log n)$ operations in $\mathbb{F}_q$.

To construct the polynomial representation for a and b, their permuted representations are preceded by zero and P_{n+1} is multiplied by the resulting vectors. The structure of the inverse of P_{n+1}, which we denote by L_{n+1}, is easier to describe. Hence we define a candidate for L_{n+1}. This matrix can be used to convert from polynomial to the extended permuted normal representation, i.e., it satisfies

$$\begin{aligned}&(1, \beta+\beta^{-1}, \beta^2+\beta^{-2}, \cdots, \beta^n+\beta^{-n})L_{n+1} = \\ &(1, \beta+\beta^{-1}, (\beta+\beta^{-1})^2, \cdots, (\beta+\beta^{-1})^n).\end{aligned}$$

Furthermore L_{n+1} is invertible. Then we study its structure and exhibit a factorization into sparse factors in Section 5, which is also used to find a factorization for P_n.

Definition 1. *For integers i, j let $l_{i,j} \in \mathbb{F}_p$ be such that $(x+x^{-1})^j = \sum_{i\in\mathbb{Z}} l_{i,j}x^i$ in $\mathbb{F}_p[x]$, and $L_n = (l_{i,j})_{0\leq i,j<n} \in \mathbb{F}_p^{n\times n}$.*

Obviously $l_{i,j} = 0$ for $|i| > |j|$.

Example 1. Let $q = 9$, i.e., $p = 3$. For $0 \leq j < 9$, the powers $(x + x^{-1})^j$ and hence the matrix L_9 are:

j	$(x+x^{-1})^j$
0	1
1	$x + x^{-1}$
2	$x^2 + 2 + x^{-2}$
3	$x^3 + x^{-3}$
4	$x^4 + x^2 + x^{-2} + x^{-4}$
5	$x^5 + 2x^3 + x + x^{-1} + 2x^{-3} + x^{-5}$
6	$x^6 + 2 + x^{-6}$
7	$x^7 + x^5 + 2x + 2x^{-1} + x^{-5} + x^{-7}$
8	$x^8 + 2x^6 + x^4 + 2x^2 + 1 + 2x^{-2} + x^{-4} + 2x^{-6} + x^{-8}$

$$L_9 = \begin{pmatrix} 1&0&2&0&0&0&2&0&1 \\ 0&1&0&0&0&1&0&2&0 \\ 0&0&1&0&1&0&0&0&2 \\ 0&0&0&1&0&2&0&0&0 \\ 0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&2 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

Theorem 1. *The matrix L_n of Definition 1 satisfies*

$$\begin{array}{l} (1, \beta + \beta^{-1}, \beta^2 + \beta^{-2}, \cdots, \beta^{n-1} + \beta^{-n+1})L_n = \\ (1, \beta + \beta^{-1}, (\beta + \beta^{-1})^2, \cdots, (\beta + \beta^{-1})^{n-1}), \end{array} \tag{3}$$

is upper triangular with 1 *on the diagonal, hence nonsingular, and its entries satisfy the relation:*

$$(L_n)_{i,j} = \begin{cases} 0 & \text{if } i > j \text{ or } j - i \text{ is odd, and} \\ \binom{j}{(j-i)/2} & \text{otherwise.} \end{cases}$$

Definition 2. *We denote the inverse of L_n by $P_n = (p_{i,j})_{0 \leq i,j < n} \in \mathbb{F}_p^{n \times n}$.*

5 Factorizations of the Conversion Matrices

The cost of computing the isomorphisms π_n and ν_n of Section 3 depends on the structure of the corresponding matrices. As in the last section, it is easier to initially study the structure of L_n and use this information to analyze P_n. The former study will be simplified by assuming n to be a power of p, say $n = p^r$, and extending the results to general n later. This simplification enables a recursive representation of L_{p^r} which is exhibited in Lemma 1. This recursive structure is then used in Theorem 3 to find a factorization of L_{p^r} into sparse matrices. To describe the recursive structure of L_{p^r} we define three matrices of *reflection*, *shifting*, and *factorization* denoted by Θ_n, Ψ_n, and B_r, respectively.

Definition 3. *The entries of the* reflection *and* shifting *matrices $\Theta_n = (\theta_{i,j})_{0 \leq i,j < n} \in \mathbb{F}_p^{n \times n}$ and $\Psi_n = (\psi_{i,j})_{0 \leq i,j < n} \in \mathbb{F}_p^{n \times n}$, respectively, are defined by the relations:*

$$\theta_{i,j} = \begin{cases} 1 & \text{if } i + j = n, \\ 0 & \text{otherwise,} \end{cases} \qquad \psi_{i,j} = \begin{cases} 1 & \text{if } j - i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

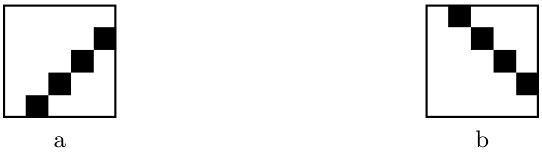


Fig. 2. (a) The matrix Θ_5 and (b) the matrix Ψ_5.

As an example, Θ_5 and Ψ_5, are shown in Figure 2, where the coefficients equal to 0 and 1 are represented by empty and filled boxes, respectively. Left multiplication by Θ_n reflects a matrix horizontally and shifts the result by one row downwards. Right multiplication by Ψ_n shifts a matrix by one position upwards.

Definition 4. *The* factorization *matrix B_r is:*

$$B_r = L_p \otimes I_{p^{r-1}} + (\Psi_p L_p) \otimes \Theta_{p^{r-1}} \in \mathbb{F}_p^{p^r \times p^r},$$

where $\otimes$ is the Kronecker or tensor product operator.

Using Definitions 1 and 4 it is easy to prove the following theorem which gives more information about the structure of B_r and can be helpful for constructing this matrix. The matrices B_3 and L_{27} are shown in Figure 3.

Theorem 2. *The matrix B_r can be split into $p \times p$ blocks $B^{(i_1,j_1)} \in \mathbb{F}_p^{p^{r-1} \times p^{r-1}}$ such that $B_r = (B^{(i_1,j_1)})_{0 \le i_1,j_1 < p}$ and*

$$B^{(i_1,j_1)} = \begin{cases} \text{the zero block} & \text{if } i_1 > j_1, \\ \binom{j_1}{(j_1-i_1)/2} I_{p^{r-1}} & \text{if } i_1 \le j_1 \text{ and } j_1 - i_1 \text{ is even, and} \\ \binom{j_1}{(j_1-i_1-1)/2} \Theta_{p^{r-1}} & \text{otherwise.} \end{cases}$$

Lemma 1. *For $r \ge 1$, we have*

$$L_{p^r} = B_r(I_p \otimes L_{p^{r-1}}). \tag{4}$$

Proof. For $0 \le i, j < p^r$ we compute $(L_{p^r})_{i,j}$ by writing

$$i = i_1 p^{r-1} + i_0,\ j = j_1 p^{r-1} + j_0, \tag{5}$$

with $0 \le i_1, j_1 < p$ and $0 \le i_0, j_0 < p^{r-1}$. Since $p \cdot x = 0$, we have

$$(x + x^{-1})^j = (x + x^{-1})^{j_1 p^{r-1}} (x + x^{-1})^{j_0} = (x^{p^{r-1}} + x^{-p^{r-1}})^{j_1} (x + x^{-1})^{j_0} = \\ (\sum_{k_1 \in \mathbb{Z}} l_{k_1,j_1} x^{k_1 p^{r-1}})(\sum_{k_0 \in \mathbb{Z}} l_{k_0,j_0} x^{k_0}) = \sum_{k_0,k_1 \in \mathbb{Z}} l_{k_1,j_1} l_{k_0,j_0} x^{k_1 p^{r-1} + k_0} \tag{6}$$

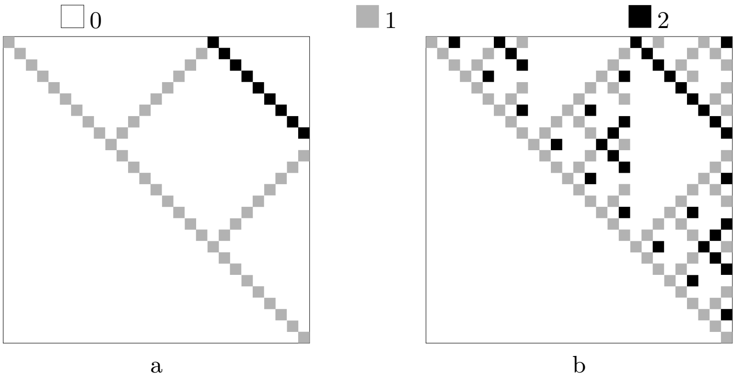


Fig. 3. (a) the matrix B_3 and (b) the matrix L_{27} for $p = 3$

where $l_{k,j}$ is as Definition 1 and is zero for $|k| > |j|$. For the coefficient of $x^i = x^{i_1 p^{r-1} + i_0}$, which is $(L_{p^r})_{i,j}$, we have:

$$\begin{aligned} &k_1 p^{r-1} + k_0 = i_1 p^{r-1} + i_0 \Longrightarrow k_0 \equiv i_0 \mod p^{r-1} \Longrightarrow \\ &k_0 = i_0 + tp^{r-1} \text{ and } k_1 = i_1 - t \text{ for some } t \in \mathbb{Z}. \end{aligned} \tag{7}$$

In the above equation except for $t = -1, 0$ we have $|i_0 + tp^{r-1}| \geq |p^{r-1}| > |j_0|$ which means $l_{i_0 + tp^{r-1}, j_0} = 0$, and hence

$$(L_{p^r})_{i,j} = l_{i_1,j_1} l_{i_0,j_0} + l_{i_1+1,j_1} l_{i_0 - p^{r-1}, j_0}, \tag{8}$$

in which $l_{i_1,j_1} = (L_p)_{i_1,j_1}$, $l_{i_0,j_0} = (L_{p^{r-1}})_{i_0,j_0}$, and $l_{i_1+1,j_1} = (\Psi_p L_p)_{i_1,j_1}$ according to the definition of Ψ_p. The value of $l_{i_0 - p^{r-1}, j_0}$ can be replaced by $l_{p^{r-1} - i_0, j_0}$ because of the symmetry of the binomial coefficients. The latter can again be replaced by $(\Theta_{p^{r-1}} L_{p^{r-1}})_{i_0,j_0}$, since for $0 < i_0 < p^{r-1}$ the only nonzero entry in the i_0th row of $\Theta_{p^{r-1}}$ is in the $(p^{r-1} - i_0)$th column and hence $(\Theta_{p^{r-1}} L_{p^{r-1}})_{i_0,j_0}$ is the entry in the $(p^{r-1} - i_0)$th row and j_0th column of $L_{p^{r-1}}$. For $i_0 = 0$ the entry $(\Theta_{p^{r-1}} L_{p^{r-1}})_{i_0,j_0}$ is zero since there is no nonzero entry in the i_0th row of $\Theta_{p^{r-1}}$, and l_{p^{r-1}, j_0} is also zero since $j_0 < p^{r-1}$. Substituting all of these into (8) we have

$$(L_{p^r})_{i,j} = (L_p)_{i_1,j_1} (L_{p^{r-1}})_{i_0,j_0} + (\Psi_p L_p)_{i_1,j_1} (\Theta_{p^{r-1}} L_{p^{r-1}})_{i_0,j_0} \tag{9}$$

which together with (5) shows that:

$$L_{p^r} = L_p \otimes L_{p^{r-1}} + (\Psi_p L_p) \otimes (\Theta_{p^{r-1}} L_{p^{r-1}}). \tag{10}$$

It is straightforward, using Definition 4, to show that (10) is equivalent to (4). □

This recursive relation resembles that for the DFT matrix in Chapter 1 of van Loan (1992) and enables us to find a matrix factorization for L_{p^r} in Theorem 3. Using this factorization the map of a vector under the isomorphism ν_n can be computed using $O(n \log n)$ operations as will be shown later in Section 6.

Theorem 3. *For $r \geq 1$, we have*

$$L_{p^r} = (I_1 \otimes B_r)(I_p \otimes B_{r-1}) \cdots (I_{p^{r-2}} \otimes B_2)(I_{p^{r-1}} \otimes B_1). \tag{11}$$

In order to multiply L_{p^r} by a vector, we successively multiply the matrices in the factorization (11) by that vector. In the next section we count the number of operations required for the computations of the mappings π_n and ν_n

6 Cost of computing ν_n and π_n

Multiplication by L_{p^r} consists of several multiplications by B_k for different values of k. Hence it is better to start the study by counting the required operations for multiplying B_k by a vector in $\mathbb{F}_q^{p^k}$. The number of nonzero entries in the matrices L_p, $\psi_p L_p$, $I_{p^{k-1}}$, and $\Theta_{p^{k-1}}$ are dominated by $p^2/4$, $p^2/4$, p^{k-1}, and p^{k-1}, respectively. Hence using Definition 4, we expect the number of operations to multiply B_k by a vector in $\mathbb{F}_q^{p^k}$ be a polynomial in p, dominated by $p^{k+1}/2$. A more accurate expression for this cost is given in Lemma 2.

Definition 5. *We define $\mu_{add}(k)$ and $\mu_{mult}(k)$ to be the number of additions and multiplications, respectively, in $\mathbb{F}_q$ to multiply B_k by a vector in $\mathbb{F}_q^{p^k}$. We define further δ by the relation*

$$\delta = \begin{cases} 1 \text{ if } p = 2, \\ 0 \text{ otherwise.} \end{cases}$$

Lemma 2. *For $k \geq 1$, we have*

$$\begin{aligned} \mu_{add}(k) &\leq (p-1)(2p^k - p - 1)/4 - \delta/4, \\ \mu_{mult}(k) &\leq (1-\delta) \cdot \mu_{add}(k). \end{aligned}$$

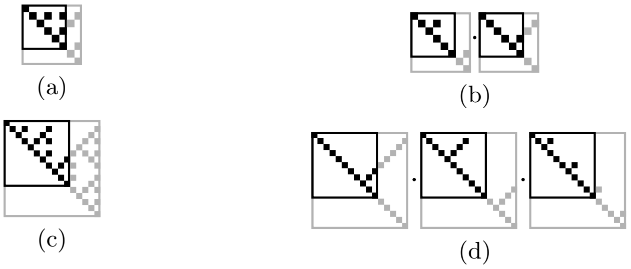
For this estimate we use information about the structural zeros in B_{k-1} according to Theorem 2, but ignore the fact that some binomial coefficients might vanish modulo p. As an example since B_1, for $p = 2$, is the identity matrix both $\mu_{add}(1)$ and $\mu_{mult}(1)$ are zero.

Using Lemma 2 and Theorem 3 we are now in the position to estimate the cost of multiplication by L_{p^r}.

Lemma 3. *Multiplying L_{p^r} by a vector in $\mathbb{F}_q^{p^r}$ for $r \geq 1$ requires at most $\eta(r)$ additions, where*

$$\eta(r) = r(p-1)p^r/2 - (p+1)(p^r - 1)/4 - \delta(2^r - 1)/4.$$

The number of multiplications is not larger than the number of additions.


(a)

(b)

(c)

(d)

Fig. 4. (a) The matrices P_6 (black) and P_8 (gray), (b) their factorizations, (c) the matrices L_{11} (black) and L_{16} (gray), and (d) their factorizations for $p = 2$.

The following theorem is an application of Lemma 3, using $r = \lceil \log_p(n+1) \rceil$.

Theorem 4. *Multiplication of L_n by a vector in $\mathbb{F}_q^n$ can be done using $O(n \log n)$ operations in $\mathbb{F}_q$.*

We observe that each B_r is nonsingular since it is upper triangular and all of the entries on the main diagonal are 1. Using (11), the matrix P_{p^r} can be factored as:

$$P_{p^r} = L_{p^r}^{-1} = (I_{p^{r-1}} \otimes B_1^{-1})(I_{p^{r-2}} \otimes B_2^{-1}) \cdots (I_p \otimes B_{r-1}^{-1})(I_1 \otimes B_r^{-1}). \qquad (12)$$

Studying the structure of B_r^{-1} in terms of $\Theta_{p^{r-1}}$ and $I_{p^{r-1}}$ reveals that B_r^{-1} does not have more nonzero entries than B_r. We omit the complete proof here for sake of brevity and refer the reader to the full paper or Section 4.6 of Shokrollahi (2006).

Theorem 5. *Multiplication of P_n from Definition 2 by a vector in $\mathbb{F}_q^n$ can be done using $O(n \log n)$ operations in $\mathbb{F}_q$.*

The matrices L_{11} and P_6 when $p = 2$ and their factorizations are shown in Figure 4. We conclude this section with the following theorem. Although its result is not concerned with normal basis multiplication directly, it emphasizes the most important property of our multiplier. Namely a specific change of basis in $\mathbb{F}_{q^n}$ which can be done using $O(n \log n)$ instead of $O(n^2)$ operations, which is the cost of general basis conversion in $\mathbb{F}_{q^n}$.

Theorem 6. *Let $\mathcal{N}$ be a type 2 normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ generated by the normal element $\beta + \beta^{-1}$ and*

$$\mathcal{P} = (1, \beta + \beta^{-1}, \cdots, (\beta + \beta^{-1})^{n-1})$$

be the polynomial basis generated by the minimal polynomial of $\beta + \beta^{-1}$. Then the change of representation between the two bases $\mathcal{N}$ and $\mathcal{P}$ can be done using $O(n \log n)$ operations in $\mathbb{F}_q$.

7 Other Costs

There are two other operations in our multiplier, namely polynomial multiplication and conversion from the extended permuted representation to the normal basis representation.

The polynomial multiplication method can be selected arbitrarily among all available methods depending on the polynomial lengths and the implementation environments. Another cost is the number of bit operations to convert from extended permuted to the permuted representation. By multiplying the polynomials of length $n+1$, the product which is of length $2n+1$ is converted to a linear combination of $\beta^i + \beta^{-i}$ for $0 \le i \le 2n$. These values should be converted to the permuted representation, i.e., $\beta^i + \beta^{-i}$ for $1 \le i \le n$. This conversion is done using the fact that β is a $2n+1$st root of unity. The cost for the case of odd prime numbers is given in the next theorem.

Theorem 7. *Let q be odd. Conversion from the extended permuted representation of the product in Figure 1 into the permuted basis can be done using at most $2n$ additions and n scalar multiplications in $\mathbb{F}_q$.*

When $p = 2$, the constant term vanishes because of Lucas' theorem, and the above task requires only n additions. Using the material presented herein we can summarize the costs of our multiplier in the following theorem. Since we can use any suitable polynomial multiplier, the cost depends on the polynomial multiplication method used.

Theorem 8. *Let $\mathbb{F}_{q^n}$ be a finite field of characteristic p, which contains an optimal normal basis of type 2 over $\mathbb{F}_q$. Multiplication in this normal basis can be done using at most*

$$n + 2(1-\delta)n + 2\eta(r_1) + \eta(r_2) + \mathsf{M}(n+1)$$

operations in $\mathbb{F}_q$, where δ is defined in Definition 5, η in Lemma 3, $\mathsf{M}(n)$ is the number of $\mathbb{F}_q$-operations to multiply two polynomials of degree $n-1$, $r_1 = \lceil \log_p(n+1) \rceil$, and $r_2 = \lceil \log_p(2n+1) \rceil$. For sufficiently large n the above expression is at most

$$\mathsf{M}(n+1) + 3n + 2(2n+1)p^2 \log_p(2n+1).$$

8 Comparison

Our multiplier is especially efficient when the extension degree n is much larger than the size of the ground field q. One practical application of this kind is cryptography in fields of characteristic 2. In this section we compare this multiplier with some other structures from the literature for this task. The field extensions which are discussed here are prime numbers n such that $\mathbb{F}_{2^n}$ contains an optimal normal basis of type 2.

The first structures which we study here are the circuits of Sunar & Koç (2001) and Reyhani-Masoleh & Hasan (2002). Both of these circuits require $n(5n-1)/2$ gates and we group them together as *classical*. The second circuit is from Gao et al. (2000). The idea behind this multiplier is to consider the representation

$$a_1(\beta+\beta^{-1})+\cdots+a_n(\beta^n+\beta^{-n})$$

as the sum of two polynomials

$$a_1\beta+\cdots+a_n\beta^n \text{ and } a_n\beta^{-n}+\cdots+a_1\beta^{-1}.$$

To multiply two elements four polynomials of degree n should be multiplied together. However, because of the symmetry only two multiplications are necessary which also yield the other two products by mirroring the coefficients. The cost of a multiplication using this circuit is $2\mathsf{M}(n)+2n$, where $\mathsf{M}(n)$ is the cost of multiplying two polynomials of length n. We may use for $\mathsf{M}(n)$ the cost of the multiplier by von zur Gathen & Shokrollahi (2005) which is not larger than $\lceil 7.6n^{\log_2 3}\rceil$ in our range.

To have a rough measure of hardware cost, we compare the circuits with respect to both area and area-time (AT). By time we mean the depth of the circuit implementation of a parallel multiplier in terms of the number of `AND` and `XOR` gates. The propagation delay of the classical multiplier is $1+\lceil\log_2 n\rceil$ gates. The propagation delay of the multiplier of this chapter consists of two parts: the first one belongs to the conversion circuits which is $2+2\lceil\log_2 n\rceil$ and the other part corresponds to the polynomial multiplier. We compute the propagation delay of each polynomial multiplier for that special case. The propagation delay of the multiplier of Gao et al. (2000) is two plus the delay of each polynomial multiplier which must again be calculated for each special case.

The area and AT parameters of these three circuits are compared with each other and the results are shown in Figure 5. Please note that the costs of our designs are exact values from Theorem 8. In these diagrams polynomial multiplication is done using the methods of von zur Gathen and Shokrollahi (2005). As it can be seen the area of the proposed multiplier is always better than the other two structures. But the AT parameter is larger for small finite fields. This shows that, as we have mentioned, this method is appropriate for applications where only small area is available, or where the finite fields are large. Economical applications where small FPGAs should be used or RFID technology, are situations of this sort. The AT parameter of the proposed multiplier is $O(n\log^3 n(\log\log n)^3)$, whereas that of the classical multiplier is $O(n^2\log n)$.

Another method which should be compared to ours is the method of Fan & Hasan (2006). This work has been introduced to us by one of the referees and we did not have enough time for an exact comparison. This method requires roughly $13n^{1.6}$ gates, whereas the Karatsuba method for polynomials of length n needs $9n^{1.6}$ gates. Hence we approximate the number of operations for this method to be $13/9\mathsf{M}(n)$ for the Karatsuba method. The delay of this method equals the delay of the Karatsuba method. We guess the number of operations for this method to be larger than ours for the given bound, but the area-time parameter must be better. We again emphasize that their methods is comparable to the Karatsuba method, whereas ours can use any asymptotically fast method like that of Cantor with a cost of $O(n \log^3 n (\log \log n)^2)$.

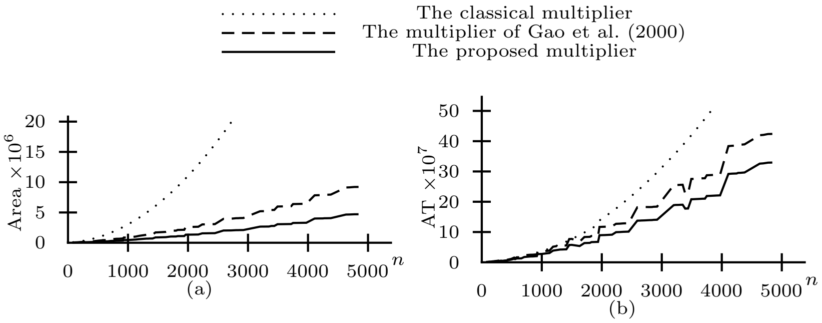


Fig. 5. Comparing the (a) area (as the number of two-input gates) and (b) the AT parameter (as the product of the number of two-input gates and the delay of a single gate) of three multipliers for binary finite fields $\mathbb{F}_{2^n}$ such that n is a prime smaller than 5000 and $\mathbb{F}_{2^n}$ contains an optimal normal basis of type 2.

9 Conclusion

This work presents a new algorithm for multiplication in finite fields using optimal normal bases of type 2 which reduces the asymptotic number of operations from $2\mathsf{M}(n) + O(n)$ reported by Gao et al. (2000) to $\mathsf{M}(n) + O(n \log n)$. The efficiency of this multiplier arises from a fast transformation between normal bases and suitable polynomial representation. This transformation can be done by $O(n \log n)$ operations instead of generic $O(n^2)$ operations for the general case. This algorithm is especially attractive for hardware implementations where area resources are limited as also shown in comparisons with other methods from the literature.

Acknowledgement

We thank anonymous referees for their helpful suggestions and also for pointing to the work of Fan & Hasan (2006).

References

1. U.S. Department of Commerce / National Institute of Standards and Technology: Digital Signature Standard (DSS). (2000) Federal Information Processings Standards Publication 186-2.

2. Fan, H., Hasan, M.A.: Subquadratic multiplication using optimal normal bases. Technical Report cacr2006-26, University of Waterloo, Waterloo (2006).
3. Gao, S., von zur Gathen, J., Panario, D., Shoup, V.: Algorithms for exponentiation in finite fields. Journal of Symbolic Computation **29** (2000) 879–889.
4. Gao, S., von zur Gathen, J., Panario, D.: Gauss periods and fast exponentiation in finite fields. In: Proc. of the Second Latin American Symposium on Theoretical Informatics (LATIN'95), Valparaiso, Chile (1995) 311–322.
5. Gao, S., Lenstra, Jr., H.W.: Optimal normal bases. Designs, Codes, and Cryptography **2** (1992) 315–323.
6. von zur Gathen, J., Nöcker, M.: Polynomial and normal bases for finite fields. Journal of Cryptology **18** (2005) 313–335.
7. von zur Gathen, J., Shokrollahi, J.: Efficient FPGA-based Karatsuba multipliers for polynomials over $\mathbb{F}_2$. In Preneel, B., Tavares, S., eds.: Selected Areas in Cryptography (SAC 2005). Number 3897 in Lecture Notes in Computer Science, Kingston, ON, Canada, Springer-Verlag (2005) 359–369.
8. Grabbe, C., Bednara, M., Shokrollahi, J., Teich, J., von zur Gathen, J.: FPGA designs of parallel high performance $GF(2^{233})$ multipliers. In: Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS-03). Volume II., Bangkok, Thailand (2003) 268–271.
9. Granger, R., Page, D., Stam, M.: Hardware and software normal basis arithmetic for pairing-based cryptogaphy in characteristic three. IEEE Transactions on Computers **54** (2005) 852–860.
10. Kaliski, B.S., Liskov, M.: Efficient Finite Field Basis Conversion Involving Dual Bases. In Koç, Ç.K., Paar, C., eds.: Cryptographic Hardware and Embedded Systems. Number 1717 in Lecture Notes in Computer Science, Springer-Verlag (1999) 135–143.
11. van Loan, C.: Computational Frameworks for the Fast Fourier Transform. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1992).
12. Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A., Wilson, R.M.: Optimal normal bases in GF(p^n). Discrete Applied Mathematics **22** (1989) 149–161.
13. Nöcker, M.: Data structures for parallel exponentiation in finite fields. Doktorarbeit, Universität Paderborn, Germany (2001).
14. Omura, J.K., Massey, J.L.: Computational method and apparatus for finite field arithmetic. United States Patent 4,587,627 (1986) Date of Patent: May 6, 1986.
15. Reyhani-Masoleh, A., Hasan, M.A.: A new construction of Massey-Omura parallel multiplier over $GF(2^m)$. IEEE Transactions on Computers **51** (2002) 511–520.
16. Shokrollahi, J.: Efficient Implementation of Elliptic Curve Cryptography on FPGAs. PhD thesis, Bonn University, Bonn (2006) http://hss.ulb.uni-bonn.de/diss_online/math_nat_fak/2007/shokrollahi_jamshid/ index.htm.
17. Sunar, B., Koç, Ç.K.: An efficient optimal normal basis type II multiplier. IEEE Transactions on Computers **50** (2001) 83–87.
18. Wassermann, A.: Konstruktion von Normalbasen. Bayreuther Math. Schriften **31** (1990) 155–164.
19. Wikipedia: Sierpinski triangle. Webpage (2006) http://en.wikipedia.org/wiki/Sierpinski_triangle.