Explicit formulas for efficient multiplication in $\mathbb{F}_{3^{6m}}$

Elisa Gorla[1], Christoph Puttmann[2], and Jamshid Shokrollahi[3]

[1] University of Zurich, Switzerland
elisa.gorla@math.unizh.ch
[2] Heinz Nixdorf Institute, University of Paderborn, Germany
puttmann@hni.upb.de
[3] Ruhr University, Bochum, Germany
jamshid@crypto.rub.de

Abstract. Efficient computation of the Tate pairing is an important part of pairing-based cryptography. Recently with the introduction of the Duursma-Lee method special attention has been given to the fields of characteristic 3. Especially multiplication in $\mathbb{F}_{3^{6m}}$, where m is prime, is an important operation in the above method. In this paper we propose a new method to reduce the number of $\mathbb{F}_{3^m}$-multiplications for multiplication in $\mathbb{F}_{3^{6m}}$ from 18 in recent implementations to 15. The method is based on the fast Fourier transform and its explicit formulas are given. The execution times of our software implementations for $\mathbb{F}_{3^{6m}}$ show the efficiency of our results.

Keywords: Finite field arithmetic, fast Fourier transform, Lagrange interpolation, Tate pairing computation

1 Introduction

Efficient multiplication in finite fields is a central task in the implementation of most public key cryptosystems. A great amount of work has been devoted to this topic (see [1] or [2] for a comprehensive list). The two types of finite fields which are mostly used in cryptographic standards are binary finite fields of type $\mathbb{F}_{2^m}$ and prime fields of type $\mathbb{F}_p$, where p is a prime (cf. [3]). Efforts to efficiently fit finite field arithmetic into commercial processors resulted into applications of medium characteristic finite fields like those reported in [4] and [5]. Medium characteristic finite fields are fields of type $\mathbb{F}_{p^m}$, where p is a prime slightly smaller than the word size of the processor, and has a special form that simplifies the modular reduction. Mersenne prime numbers constitute an example of primes which are used in this context. The security parameter is given by the length of the binary representations of the field elements, and the extension degree m is selected appropriately. Due to security considerations, the extension degree for fields of characteristic 2 or medium characteristic is usually chosen to be prime.

With the introduction of the method of Duursma and Lee for the computation of the Tate pairing (see [6]), fields of type $\mathbb{F}_{3^m}$ for m prime have attracted

special attention. Computing the Tate pairing on elliptic curves defined over $\mathbb{F}_{3^m}$ requires computations both in $\mathbb{F}_{3^m}$ and in $\mathbb{F}_{3^{6m}}$. In the paper [7], calculations are implemented using the tower of extensions

$$\mathbb{F}_{3^m} \subset \mathbb{F}_{3^{2m}} \subset \mathbb{F}_{3^{6m}}.$$

Multiplications in $\mathbb{F}_{3^{2m}}$ and $\mathbb{F}_{3^{6m}}$ are done using 3 and 6 multiplications, respectively. This requires a total 18 multiplications in $\mathbb{F}_{3^m}$. In this paper we make use of the same extension tower, using 3 multiplications in $\mathbb{F}_{3^m}$ to multiply elements in $\mathbb{F}_{3^{2m}}$. Since we represent the elements of $\mathbb{F}_{3^{6m}}$ as polynomials with coefficients in $\mathbb{F}_{3^{2m}}$, we can use Lagrange interpolation to perform the multiplication. This requires only 5 multiplications in $\mathbb{F}_{3^{2m}}$, thus reducing the total number of $\mathbb{F}_{3^m}$ multiplications from 18 to 15. The method that we propose has a slightly increased number of additions in comparison to the Karatsuba method. Notice however that for $m > 90$ (which is the range used in the cryptographic applications) a multiplication in $\mathbb{F}_{3^m}$ requires many more resources than an addition, therefore the overall resource consumption is reduced, as also shown by the results of our software experiments shown in Section 4.

In comparison to the classical multiplication method, the Karatsuba method (see [8], [9], and [7]) reduces the number of multiplications while introducing extra additions. Since the cost of addition grows linearly in the length of the polynomials, when the degree of the field extension gets larger multiplication will be more expensive than addition. Hence the above tradeoff makes sense. The negligibility of the cost of addition compared to that of multiplication has gone so far that the theory of multiplicative complexity of bilinear maps, especially polynomial multiplication, takes into account only the number of variable multiplications (see e.g. [10] and [11]). Obviously this theoretical model is of practical interest only when the number of additions and the costs of scalar multiplications can be kept small. A famous result in the theory of multiplicative complexity establishes a lower bound of $2n + 1$ for the number of variable multiplications needed for the computation of the product of two polynomials of degree at most n. This lower bound can be achieved only when the field contains enough elements (see [12] or [13]). The proof of the theorem uses Lagrange evaluation-interpolation, which is also at the core of our approach. This is similar to the short polynomial multiplication (convolution) methods for complex or real numbers in [14]. In order for this method to be especially efficient, the points at which evaluation and interpolations are done are selected as primitive $(2n + 1)$st roots of unity. In a field of type $\mathbb{F}_{3^{2m}}$, fifth roots of unity do not exist for odd m. We overcome this problem by using fourth roots of unity instead. Notice that a primitive fourth root of unity always exist in a field of type $\mathbb{F}_{3^{2m}}$. We use an extra point to compute the fifth coefficient of the product. An advantage of using a primitive fourth root of unity is that the corresponding interpolation matrix will be a 4×4 DFT matrix, and the evaluations and interpolations can be computed using radix-2 FFT techniques (see [15] or [16]) to save some further number of additions and scalar multiplications. The current work can be considered as the continuation of that in [17] for combination of the linear-time

multiplication methods with the classical or Karatsuba ones to achieve efficient polynomial multiplication formulas.

Our work is organized as follows. Section 2 is devoted to explaining how evaluation-interpolation can be used in general to produce short polynomial multiplication methods. In Section 3 we show how to apply this method to our special case, and produce explicit formulas for multiplication of polynomials of degree at most 2 over $\mathbb{F}_{3^{2m}}$. In Section 4 we fine-tune our method using FFT techniques, and give timing results of software implementations and also explicit multiplication formulas. Section 5 shows how our results can be used in conjunction with the method of Duursma-Lee for computing the Tate pairing on some elliptic and hyperelliptic curves. Section 6 contains some final remarks and conclusions.

2 Multiplication using evaluation and interpolation

We now explain the Lagrange evaluation-interpolation for polynomials with coefficients in $\mathbb{F}_{p^m}$. Throughout this section m is not assumed to be prime (in the next section we will replace m by $2m$). Let

$$a(z) = a_0 + a_1 z + \cdots + a_n z^n \in \mathbb{F}_{p^m}[z]$$
$$b(z) = a_0 + a_1 z + \cdots + a_n z^n \in \mathbb{F}_{p^m}[z]$$

be given such that

$$p^m > 2n. \tag{1}$$

We represent the product of the two polynomials by

$$c(z) = a(z)b(z) = c_0 + c_1 z + \cdots + c_{2n} z^{2n}$$

and let $e = (e_0, \cdots, e_{2n}) \in \mathbb{F}_{p^m}^{2n+1}$ be a vector with $2n+1$ distinct entries. Evaluation at these points is given by the map ϕ_e

$$\phi_e : \mathbb{F}_{p^m}[z] \to \mathbb{F}_{p^m}^{2n+1}$$
$$\phi_e(f) = (f(e_0), \cdots, f(e_{2n})).$$

Let $A, B, C \in \mathbb{F}_{p^m}^{2n+1}$ denote the vectors $(a_0, \cdots, a_n, 0, \cdots, 0)$, $(b_0, \cdots, b_n, 0, \cdots, 0)$, and $(c_0, \cdots, c_{2n})$, respectively. Using the above notation we have

$$\phi_e(a) = V_e A^T, \quad \phi_e(b) = V_e B^T, \quad \text{and} \quad \phi_e(c) = V_e C^T,$$

where V_e is the Vandermonde matrix

$$V_e = \begin{pmatrix} 1 & e_0 & \cdots & e_0^{2n} \\ 1 & e_1 & \cdots & e_1^{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e_{2n} & \cdots & e_{2n}^{2n} \end{pmatrix}.$$

The $2n+1$ coefficients of the product $c(z) = a(z) \cdot b(z)$ can be computed using interpolation applied to the evaluations of $c(z)$ at the chosen $2n+1$ (distinct) points of $\mathbb{F}_{p^m}$. These evaluations can be computed by multiplying the evaluations of $a(z)$ and $b(z)$ at these points. This can be formally written as

$$\phi_e(c) = \phi_e(a) * \phi_e(b)$$

where we denote componentwise multiplication of vectors by $*$. Equivalently, if we let W_e be the inverse of the matrix V_e, we have that

$$C^T = W_e(\phi_e(a) * \phi_e(b))$$

which allows us to compute the vector C, whose entries are the coefficients of the polynomial $c(z)$.

When condition (1) is satisfied, the polynomial multiplication methods constructed in this way have the smallest multiplicative complexity, i.e. the number of variable multiplications in $\mathbb{F}_{p^m}$ achieves the lower bound $2n+1$ (see [12]). Indeed (1) can be relaxed to hold even for $p^m = 2n$. In this case, a virtual element ∞ is added to the finite field. This corresponds to the fact that the leading coefficient of the product is the product of the leading coefficients of the factors.

Application of this method to practical situations is not straightforward, since the number of additions increases and eventually dominates the reduction in the number of multiplications. In order for this method to be efficient, n must be much smaller than p^m. An instance of this occurs when computing in extensions of medium size primes (see e.g. [13]). The case of small values of p is more complicated, even for small values of n. We recall that in this case the entries of the matrix V_e are in $\mathbb{F}_{p^m}$ and are generally represented as polynomials of length $m-1$ over $\mathbb{F}_p$. For multiplication of V_e by vectors to be efficient, the entries of this matrix must be chosen to be sparse. However, this gives no control on the sparsity of the entries of W_e. Indeed one requirement for the entries of W_e, in the basis $\mathcal{B}$, to be sparse is that the inverse of the determinant of V_e, namely

$$\prod_{0 \le i,j \le 2n, i \ne j} (e_i - e_j)$$

has a sparse representation in $\mathcal{B}$. We are not aware of any method which can be used here. On the other hand, it is known that if the e_i's are the elements of the geometric progression ω^i, $0 \le i \le 2n$, and ω is a $(2n+1)$st primitive root of unity, then the inverse W_e equals $1/(2n+1)$ times the Vandermonde matrix whose e_i's are the elements of the geometric progression of ω^{-1} (see [2]). We denote these two matrices by V_ω and $V_{\omega^{-1}}$, respectively. The above fact suggests that choosing powers of roots of unity as interpolation points should enable us to control the sparsity of the entries of the corresponding Vandermonde matrix. Roots of unity are used in different contexts for multiplication of polynomials, e.g. in the FFT (see [2]) or for the construction of short multiplication methods in [14]. In the next section we discuss how to use fourth roots of unity to compute multiplication in $\mathbb{F}_{p^{6m}}$, using only 5 multiplications in $\mathbb{F}_{3^{2m}}$.

3 Multiplication using roots of unity

Elements of $\mathbb{F}_{3^{6m}}$ can be represented as polynomials of degree at most 2 over $\mathbb{F}_{3^{2m}}$. Therefore, their product is given by a polynomial of degree at most 4 with coefficients in $\mathbb{F}_{3^{2m}}$. In order to use the classical evaluation-interpolation method we would need a primitive fifth root of unity. This would require $3^{2m}-1$ to be a multiple of 5, and this is never the case unless m is even (recall that cryptographic applications require m to be prime). However using the relation

$$c_4 = a_2 b_2 \tag{2}$$

we can compute the coefficients of $c(x)$ via

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a(1)b(1) - c_4 \\ a(\omega)b(\omega) - c_4 \\ a(\omega^2)b(\omega^2) - c_4 \\ a(\omega^3)b(\omega^3) - c_4 \end{pmatrix} \tag{3}$$

where ω is a fourth root of unity. Now we apply (2) and (3) to find explicit formulas for multiplying two polynomials of degree at most 2 over $\mathbb{F}_{3^{2m}}$, where $m > 2$ is a prime.

We follow the tower representation of [7], i.e.

$$\begin{aligned} \mathbb{F}_{3^m} &\cong \mathbb{F}_3[x]/(f(x)) \\ \mathbb{F}_{3^{2m}} &\cong \mathbb{F}_{3^m}[y]/(y^2+1) \end{aligned} \tag{4}$$

where $f(x) \in \mathbb{F}_3[x]$ is an irreducible polynomial of degree m. Denote by s the equivalence class of y. Note that for odd $m > 2$, $4 \not| 3^m - 1$ and hence y^2+1 is irreducible over $\mathbb{F}_{3^m}$ since the roots of y^2+1 are fourth roots of unity. Let

$$a(z) = a_0 + a_1 z + a_2 z^2, \quad b(z) = b_0 + b_1 z + b_2 z^2 \tag{5}$$

be polynomials in $\mathbb{F}_{p^{3^{2m}}}[z]^{\leq 2}$. Our goal is computing the coefficients of the polynomial

$$c(z) = a(z)b(z) = c_0 + c_1 z + \cdots c_4 z^4.$$

Evaluation of $a(z)$ and $b(z)$ at $(1, s, s^2, s^3) = (1, s, -1, -s)$ can be done by multiplying the Vandermonde matrix of powers of s

$$V_s = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & s & -1 & -s \\ 1 & -1 & 1 & -1 \\ 1 & -s & -1 & s \end{pmatrix} \tag{6}$$

by the vectors $(a_0, a_1, a_2, 0)^T$ and $(b_0, b_1, b_2, 0)^T$, respectively. This yields the vectors

$$\phi_e(a) = \begin{pmatrix} a_0 + a_1 + a_2 \\ a_0 + sa_1 - a_2 \\ a_0 - a_1 + a_2 \\ a_0 - sa_1 - a_2 \end{pmatrix} \quad \text{and} \quad \phi_e(b) = \begin{pmatrix} b_0 + b_1 + b_2 \\ b_0 + sb_1 - b_2 \\ b_0 - b_1 + b_2 \\ b_0 - sb_1 - b_2 \end{pmatrix}.$$

Let $\phi_e(c) = \phi_e(a) * \phi_e(b)$ be the componentwise product of $\phi_e(a)$ and $\phi_e(b)$

$$\phi_e(c) = \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} (a_0 + a_1 + a_2)(b_0 + b_1 + b_2) \\ (a_0 + sa_1 - a_2)(b_0 + sb_1 - b_2) \\ (a_0 - a_1 + a_2)(b_0 - b_1 + b_2) \\ (a_0 - sa_1 - a_2)(b_0 - sb_1 - b_2) \end{pmatrix}.$$

Using (2) and (3) we get

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = W_s \begin{pmatrix} P_0 - P_4 \\ P_1 - P_4 \\ P_2 - P_4 \\ P_3 - P_4 \end{pmatrix},$$

where $P_4 = a_2 b_2$ and

$$W_s = V_s^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -s & -1 & s \\ 1 & -1 & 1 & -1 \\ 1 & s & -1 & -s \end{pmatrix} \tag{7}$$

Thus the explicit formulas for the coefficients of the product are

$$\begin{aligned} c_0 &= P_0 + P_1 + P_2 + P_3 - P_4 \\ c_1 &= P_0 - sP_1 - P_2 + sP_3 \\ c_2 &= P_0 - P_1 + P_2 - P_3 \\ c_3 &= P_0 + sP_1 - P_2 - sP_3 \\ c_4 &= P_4. \end{aligned} \tag{8}$$

4 Efficient implementation

We owe the efficiency of our method to the Cooley-Tukey factorization of the DFT matrix ([15]). The matrices V_s and W_s in (6) and (7) are not sparse, but they are the DFT matrices of the fourth roots of unity s and s^3, respectively. Hence they can be factored as a product of two sparse matrices as shown in (9) and (10).

$$V_s = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & s & -1 & -s \\ 1 & -1 & 1 & -1 \\ 1 & -s & -1 & s \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & s \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -s \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}, \tag{9}$$

$$W_s = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -s & -1 & s \\ 1 & -1 & 1 & -1 \\ 1 & s & -1 & -s \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -s \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & s \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}. \tag{10}$$

The factorizations in (9) and (10) allow us to efficiently compute the product of the matrices V_s and W_s with vectors. Notice also that the product of an element

$\omega = us + v \in \mathbb{F}_{3^m}[s]^{\leq 1} \cong \mathbb{F}_{3^{2m}}$ with s equals $vs - u$. Hence multiplying by s an element of $\mathbb{F}_{3^{2m}}$ is not more expensive than a change of sign.

Notice that in alternative to the Vandermonde matrix corresponding to s we could use the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & s & -1 & -s \end{pmatrix}$$

whose inverse is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ s & 1-s & -1-s & s \\ -1 & -1 & -1 & 0 \\ -s & 1+s & -1+s & -s \end{pmatrix}.$$

Obviously the latter matrices are sparse but since they do not possess any special structure up to our knowledge, multiplying them by vectors is more expensive than multiplying V_s and W_s.

Multiplying elements in the field $\mathbb{F}_{3^{6\cdot 97}}$ is required in the Tate pairing computation on the group of $\mathbb{F}_{3^{97}}$-rational points of the elliptic curves

$$E_d : y^2 = x^3 - x + d \quad d \in \{-1, 1\}$$

defined over $\mathbb{F}_3$. An efficient algorithm for the computation of the Tate pairings on these curves is discussed in [6].

We have implemented the multiplication over $\mathbb{F}_{3^{6\cdot 97}}$ using the Karatsuba method, the Montgomery method from [18], and our proposed method on a PC with an AMD Athlon 64 processor 3500+. The processor was running at 2.20 GHz and we have used the NTL library (see [19]) for multiplication in $\mathbb{F}_{3^{97}}$. Please note that although we have chosen $m = 97$ for benchmarking purposes, these methods can be applied to any odd $m > 2$ as mentioned in Section 3.

Multiplication method	**Elapsed time (ms)**
Karatsuba method	1.698
Montgomery method	1.605
Proposed method	1.451

Table 1. Comparison of the execution times of the Karatsuba and Montgomery multipliers with the proposed method for $\mathbb{F}_{3^{6m}}$.

The execution times are shown in Table 1. For the Karatsuba and the proposed methods we have used the tower of extensions

$$\mathbb{F}_{3^{97}} \subset \mathbb{F}_{3^{2\cdot 97}} \subset \mathbb{F}_{3^{6\cdot 97}},$$

where

$$\mathbb{F}_{3^{97}} \cong \mathbb{F}_3[x]/(x^{97} + x^{16} + 2)$$
$$\mathbb{F}_{3^{2\cdot 97}} \cong \mathbb{F}_{3^{97}}[y]/(y^2 + 1)$$
$$\mathbb{F}_{3^{6\cdot 97}} \cong \mathbb{F}_{3^{2\cdot 97}}[z]/(z^3 - z - 1),$$

whereas for the Montgomery method the representation

$$\mathbb{F}_{3^{6\cdot 97}} \cong \mathbb{F}_{3^{97}}[y]/(y^6 + y - 1)$$

has been used. Our implementations show that the new method is almost 14% faster than the Karatsuba and 10% faster than the Montgomery method, which is almost the ratio of saved multiplications. This provides further evidence for the fact that the number of multiplications in $\mathbb{F}_{3^{97}}$ is a good indicator of the performance of the method for $\mathbb{F}_{3^{6\cdot 97}}$.

Our multiplications are based on the following formulas. Let $\alpha, \beta \in \mathbb{F}_{3^{6\cdot m}}$ be given as:

$$\alpha = a_0 + a_1 s + a_2 r + a_3 rs + a_4 r^2 + a_5 r^2 s,$$
$$\beta = b_0 + b_1 s + b_2 r + b_3 rs + b_4 r^2 + b_5 r^2 s,$$

where $a_0, \cdots, b_5 \in \mathbb{F}_{3^m}$ and $s \in F_3^{2\cdot m}$, $r \in \mathbb{F}_3^{6\cdot m}$ are roots of y^2+1 and z^3-z-1, respectively. Let their product $\gamma = \alpha\beta \in \mathbb{F}_{3^{6\cdot m}}$ be

$$\gamma = c_0 + c_1 s + c_2 r + c_3 rs + c_4 r^2 + c_5 r^2 s.$$

The coefficients c_i, for $0 \leq i \leq 5$ are computed using:

$$\begin{aligned}
&P_0 = (a_0 + a_2 + a_4)(b_0 + b_2 + b_4)\\
&P_1 = (a_0 + a_1 + a_2 + a_3 + a_4 + a_5)(b_0 + b_1 + b_2 + b_3 + b_4 + b_5)\\
&P_2 = (a_1 + a_3 + a_5)(b_1 + b_3 + b_5)\\
&P_3 = (a_0 - a_3 - a_4)(b_0 - b_3 - b_4)\\
&P_4 = (a_0 + a_1 + a_2 - a_3 - a_4 - a_5)(b_0 + b_1 + b_2 - b_3 - b_4 - b_5)\\
&P_5 = (a_1 + a_2 - a_5)(b_1 + b_2 - b_5)\\
&P_6 = (a_0 - a_2 + a_4)(b_0 - b_2 + b_4)\\
&P_7 = (a_0 + a_1 - a_2 - a_3 + a_4 + a_5)(b_0 + b_1 - b_2 - b_3 + b_4 + b_5)\\
&P_8 = (a_1 - a_3 + a_5)(b_1 - b_3 + b_5)\\
&P_9 = (a_0 - a_3 - a_4)(b_0 + b_3 - b_4)\\
&P_{10} = (a_0 + a_1 - a_2 + a_3 - a_4 - a_5)(b_0 + b_1 - b_2 + b_3 - b_4 - b_5)\\
&P_{11} = (a_1 - a_2 - a_5)(b_1 - b_2 - b_5)\\
&P_{12} = a_4 b_4\\
&P_{13} = (a_4 + a_5)(b_4 + b_5)\\
&P_{14} = a_5 b_5\\
&c_0 = -P_0 + P_2 - P_3 - P_4 + P_{10} + P_{11} - P_{12} + P_{14};\\
&c_1 = P_0 - P_1 + P_2 + P_4 + P_5 + P_9 + P_{10} + P_{12} - P_{13} + P_{14}\\
&c_2 = -P_0 + P_2 + P_6 - P_8 + P_{12} - P_{14}\\
&c_3 = P_0 - P_1 + P_2 - P_6 + P_7 - P_8 - P_{12} + P_{13} - P_{14}\\
&c_4 = P_0 - P_2 - P_3 + P_5 + P_6 - P_8 - P_9 + P_{11} + P_{12} - P_{14}\\
&c_5 = -P_0 + P_1 - P_2 + P_3 - P_4 + P_5 - P_6 + P_7 - P_8 + P_9 - P_{10} +\\
&P_{11} - P_{12} + P_{13} - P_{14}
\end{aligned}$$

5 Other applications of the proposed method

Consider the family of hyperelliptic curves

$$C_d : \; y^2 = x^p - x + d \quad d \in \{-1, 1\} \tag{11}$$

defined over $\mathbb{F}_p$, for $p = 3$ mod. 4. Let m be such that $(2p, m) = 1$ (in practice m will often be prime), and consider the $\mathbb{F}_{p^m}$-rational points of the Jacobian of C_d. An efficient implementation of the Tate pairing on these groups is given by Duursma and Lee in [6] and [20], where they extend analogous results of Barreto et. al. and of Galbraith et. al. for the case $p = 3$. Notice that this family of curves includes the elliptic curves E_d that we mentioned in the last section. In the aforementioned papers it is also shown that the curve C_d has embedding degree $2p$. In order to compute the Tate pairing on this curve, one works with the tower of field extensions

$$\mathbb{F}_{p^m} \subset \mathbb{F}_{p^{2m}} \subset \mathbb{F}_{p^{2pm}}$$

where the fields are represented as

$$\mathbb{F}_{p^{2m}} \cong \mathbb{F}_{p^m}[y]/(y^2 + 1) \quad \text{and} \quad \mathbb{F}_{p^{2pm}} \cong \mathbb{F}_{p^{2m}}[z]/(z^p - z + 2d).$$

Let $a(z), b(z) \in \mathbb{F}_{p^{2pm}}[z]^{\leq p-1}$,

$$a(z) = a_0 + a_1 z + \ldots + a_{p-1} z^{p-1},$$

$$b(z) = b_0 + b_1 z + \ldots + b_{p-1} z^{p-1}.$$

Then $c(z) = a(z)b(z)$ has $2p - 1$ coefficients, two of which can be computed as

$$c_0 = a_0 b_0 \quad \text{and} \quad c_{2(p-1)} = a_{2(p-1)} b_{2(p-1)}.$$

In order to determine the remaining $2p - 3$ coefficients, we can write a Vandermonde matrix with entries in $\mathbb{F}^*_{p^{2m}}$ using, e.g., the elements

$$1, 2, \ldots, p-1, \pm s, \ldots, \pm\frac{p-3}{2}s, \frac{p-1}{2}s.$$

Another option is writing a Vandermonde matrix using a primitive $2(p-1)$-st root of unity combined with the relation:

$$c_{2(p-1)} = a_{2(p-1)} b_{2(p-1)}.$$

Notice that there is an element of order $2(p-1)$ in $\mathbb{F}_{p^2}$, since $2(p-1)|p^2 - 1$. If a is a primitive element in $\mathbb{F}_{p^2}$, then $\omega = a^{(p+1)/2}$ is a primitive $2(p-1)$st root of unity.

6 Conclusion

In this paper we derived new formulas for multiplication in $\mathbb{F}_{3^{6m}}$, which use only 15 multiplications in $\mathbb{F}_{3^m}$. Being able to efficiently multiply elements in $\mathbb{F}_{3^{6m}}$ is a central task for the computation of the Tate pairing on elliptic and hyperelliptic curves. Our method is based on the fast Fourier transform, slightly modified to be adapted to the finite fields that we work on. Our software experiments show that this method is at least 10% faster than other proposed methods in the literature. We have also discussed use of these ideas in conjunction with the general methods of Duursma-Lee for Tate pairing computations on elliptic and hyperelliptic curves.

Acknowledgement

The research described in this paper was funded in part by the Swiss National Science Foundation, registered there under grant number 107887, and by the German Research Foundation (Deutsche Forschungsgemeinschaft DFG) under project RU 477/8. We thank also the reviewers for their precise comments.

References

1. Knuth, D.E.: The Art of Computer Programming, vol. 2, Seminumerical Algorithms. 3rd edn. Addison-Wesley, Reading MA (1998) First edition 1969.
2. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Second edn. Cambridge University Press, Cambridge, UK (2003) First edition 1999.
3. U.S. Department of Commerce / National Institute of Standards and Technology: Digital Signature Standard (DSS). (January 2000) Federal Information Processings Standards Publication 186-2.
4. Bailey, D.V., Paar, C.: Optimal extension fields for fast arithmetic in public-key algorithms. In Krawczyk, H., ed.: Advances in Cryptology: Proceedings of CRYPTO '98, Santa Barbara CA. Volume 1462 of Lecture Notes in Computer Science., Springer-Verlag (1998) 472–485
5. Avanzi, R.M., Mihăilescu, P.: Generic efficient arithmetic algorithms for PAFFs (processor adequate finite fields) and related algebraic structures (extended abstract). In: Selected Areas in Cryptography (SAC 2003), Springer-Verlag (2003) 320–334
6. Duursma, I., Lee, H.: Tate-pairing implementations for tripartite key agreement
7. Kerins, T., Marnane, W.P., Popovici, E.M., Barreto, P.S.L.M.: Efficient hardware for the tate pairing calculation in characteristic three. In: Cryptographic Hardware and Embedded Systems, CHES2005. Volume 3659 of Lecture Notes in Computer Science., Springer-Verlag (2005) 412–426
8. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics–Doklady **7**(7) (January 1963) 595–596 translated from Doklady Akademii Nauk SSSR, Vol. 145, No. 2, pp. 293–294, July, 1962.
9. Paar, C.: Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. PhD thesis, Institute for Experimental Mathematics, University of Essen, Essen, Germany (June 1994)

10. Lempel, A., Winograd, S.: A new approach to error-correcting codes. IEEE Transactions on Information Theory **IT-23** (1977) 503–508
11. Winograd, S.: Arithmetic Complexity of computations. Volume 33. SIAM, Philadelphia (1980)
12. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Number 315 in Grundlehren der mathematischen Wissenschaften. Springer-Verlag (1997)
13. Bajard, J.C., Imbert, L., Negre, C.: Arithmetic operations in finite fields of medium prime characteristic using the lagrange representation. IEEE Transactions on Computers **55**(9) (September 2006) 1167–1177
14. Blahut, R.E.: Fast Algorithms for Digital Signal Processing. Addison-Wesley, Reading MA (1985)
15. Cooley, J.W., Tukey, J.W.: An algorithm for the machine computation of the complex Fourier series. Mathematics of Computation **19** (1965) 297–31
16. Loan, C.V.: Computational Frameworks for the Fast Fourier Transform. Society for Industrial and Applied Mathematics (siam), Philadelphia (1992)
17. 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). Volume 3897 of Lecture Notes in Computer Science., Kingston, ON, Canada, Springer-Verlag (August 2005) 359–369
18. Montgomery, P.L.: Five, Six, and seven-Term Karatsuba-Like Formulae. IEEE Transactions on Computers **54**(3) (March 2005) 362–369
19. Shoup, V.: NTL: A library for doing number theory, `http://www.shoup.net/ntl`
20. Duursma, I., Lee, H.S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Advances in cryptology—ASIACRYPT 2003. Volume 2894 of Lecture Notes in Computer Science. Springer, Berlin (2003) 111–123