Efficient multiplication in characteristic three fields

Murat Cenk and M. Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
mcenk@uwaterloo.ca
ahasan@uwaterloo.ca

Abstract. Characteristic three fields denoted by $\mathbb{F}_{3^n}$, where $n \geq 1$, are used in curve based cryptography. In this paper, first we improve the well known Karatsuba 2-way and 3-way algorithms for characteristic three fields. Then, we derive a 3-way polynomial multiplication algorithm with five 1/3 size multiplications using interpolation in $\mathbb{F}_9$. After computing the arithmetic and delay complexities of the proposed algorithm, we show that it gives better arithmetic complexity results than the known algorithms. More specifically, we show that the recursive use of the 3-way algorithm in the extension field yields about 14% to 22% improvements for multiplication in $\mathbb{F}_{3^{6n}}$, which is the main operation in curve based cryptographic pairing computations. To the best of our knowledge, this is the first time it is shown that the recursive use of the 3-way approach with five multiplications gives significant improvement for multiplication in characteristic three fields of cryptographic sizes.
Keywords: Polynomial multiplication, elliptic curves, pairings, characteristic three fields

1 Introduction

Multiplication in characteristic three fields, denoted by $\mathbb{F}_{3^n}$ where $n \geq 1$, is used in curve-based cryptography. There has been a considerable amount of work on this subject to speed up the multiplication [7], [14], [8], [6], [15], [10], [2], [1], [5]. One of the common ways for multiplication in $\mathbb{F}_{3^n}$ is to use polynomial basis representation, in which the elements of $\mathbb{F}_{3^n}$ are represented by polynomials of degree $(n-1)$ over $\mathbb{F}_3$, the finite field with three elements. In order to perform multiplication in $\mathbb{F}_{3^n}$, first the polynomials are multiplied and then the product polynomial is reduced modulo an irreducible polynomial of degree n over $\mathbb{F}_3$. The arithmetic cost of the reduction step is $O(n)$ operations; on the other hand the polynomial multiplication step requires $O(n^\alpha)$ operations, where $1 < \alpha \leq 2$. Therefore, the multiplication step is more costly than the reduction step, and reducing the cost of polynomial multiplication over $\mathbb{F}_3$ directly affects the cost of multiplication in $\mathbb{F}_{3^n}$.

To the the best of our knowledge, practical low arithmetic complexity polynomial multiplication schemes are essentially Karatsuba-like algorithms. For example, recursive uses of 2-way and 3-way algorithms have the total arithmetic complexity $O(n^{1.58})$ and $O(n^{1.63})$, respectively. In this paper, after giving the improved version of the 2-way and 3-way algorithms, we propose a 3-way polynomial multiplication algorithm with five multiplications by using interpolation in $\mathbb{F}_9$. Unlike the formula in [16], we show that the recursive use of the algorithm yields the arithmetic complexity of $O(n^{1.46})$. Moreover, we compute the hidden coefficients of the complexities and compare them for cryptographic applications. In addition, the delay complexities which are useful when the algorithm is mapped on to bit parallel hardware are derived. Finally, we show that the proposed algorithm has the least arithmetic complexity for multiplication in $\mathbb{F}_{3^{6n}}$, which is the most costly part of pairing computations and report about 14% to 22% improvements for the multiplication in $\mathbb{F}_{3^{6n}}$ of cryptographic sizes.

The rest of the paper is organized as follows: Notations and preliminaries used in the rest of the paper are given in Section 2. Then the known algorithms and our suggestions for their improvements are presented in Section 3. The next section includes the proposed 3-way algorithm with five multiplications of 1/3 size. In Section 5, improvements for multiplication in $\mathbb{F}_{3^{6n}}$ are given and finally some concluding remarks are made in Section 6.

2 Notations and preliminaries

In this section we give the notations used in the rest of the paper and a few basic algorithms. Throughout the paper, unless stated otherwise, we will always assume that the fields used in this paper are characteristic three. We will use the following notations:

- $M_{3,\oplus}(n)$: the number of $\mathbb{F}_3$ *additions* (or *subtractions*) required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_3$.
- $M_{3,\otimes}(n)$: the number of $\mathbb{F}_3$ *multiplications* required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_3$.
- $M_3(n)$: the number of total $\mathbb{F}_3$ operations required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_3$.
- $M_{9,\oplus}(n)$: the number of $\mathbb{F}_3$ additions (or subtractions) required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_9$.
- $M_{9,\otimes}(n)$: the number of $\mathbb{F}_3$ multiplications required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_9$.
- $M_9(n)$: the number of total $\mathbb{F}_3$ operations required for multiplication of two degree $n-1$ polynomials over $\mathbb{F}_9$.
- $D_3(n)$: the delay complexity of multiplication of two degree $n-1$ polynomials over $\mathbb{F}_3$.
- $D_9(n)$: the delay complexity of multiplication of two degree $n-1$ polynomials over $\mathbb{F}_9$.
- $D_\oplus$: the delay of one $\mathbb{F}_3$ addition (or subtraction).
- $D_\otimes$: the delay of one $\mathbb{F}_3$ multiplication.

We represent the elements of $\mathbb{F}_{3^n}$ as the polynomials over $\mathbb{F}_3$ of degree n. On the other hand, we construct $\mathbb{F}_9 \cong \mathbb{F}_3[X]/(X^2+1)$ and assume $\omega^2+1=0$, where $\omega \in \mathbb{F}_9$.

We assume that multiplication by -1 of a polynomial is free of cost. Moreover, the addition and subtraction have the same complexities. Note also that the cost of multiplication in $\mathbb{F}_9$ can be assumed to be four multiplications and two additions in $\mathbb{F}_3$ by using the following formula:$(a+b\omega)(c+d\omega) = ac-bd+(bc+ad)\omega$. On the other hand, the cost of multiplication of an element in $\mathbb{F}_9$ by ω is free of cost since $(a+b\omega)\omega = -b+a\omega$.

Throughout the paper we will use the Master theorem to solve the recursion equations. The proof of it can be found in [13].

Theorem 1. *[13] (Master theorem) Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $M(n)$ be defined on nonnegative integers by the recurrence*

$$M(n) = aM(n/b) + f(n)$$

where if n is not divisible by b, use $\lceil n/b \rceil$. Then $M(n)$ can be bounded asymptotically as follows:

1. *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $M(n) = \Theta(n^{\log_b a})$.*
2. *If $f(n) = \Theta(n^{\log_b a})$, then $M(n) = \Theta(n^{log_b a} \log_b(n))$*
3. *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $M(n) = \Theta(f(n))$.*

3 Known algorithms and their improvements

In this section we give Karatsuba 2-way and 3-way algorithms for characteristic three fields. Following the work in [3], [19] for improving those algorithms in characteristic *two*, we improve them for characteristic *three.*

Remark 1. In order to apply the recursive 2-way or 3-way algorithms, the polynomials are split into two parts for 2-way algorithms and three parts for 3-way algorithm used. If n is not divisible by two or three, we pad one or two zeros to the polynomial so that the sizes become divisible by two or three. This effects the complexity results negligibly.

3.1 Karatsuba 2-way algorithm

Let $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$. We can divide A and B into two parts as follows: $A(X) = A_0 + A_1 X^{n/2}$ and $B(X) = B_0 + B_1 X^{n/2}$ where A_i and B_i are polynomials of degree less than $n/2$. Let $C = \sum_{i=0}^{2} C_i X^{ni/2}$ be the product of A and B. The Karatsuba 2-way algorithm is the following:

$$\begin{cases} P_0 = A_0 B_1, \ \ P_1 = (A_0 + A_1)(B_0 + B_1), \ \ P_2 = A_2 B_2, \\ C = P_0 + (P_1 - P_0 - P_2)X^{n/2} + P_2 X^n. \end{cases} \tag{1}$$

The algorithm given in (1) requires three multiplications of two degree $n/2-1$ polynomials plus $4n-4$ additions. On the other hand, the delay complexity of the algorithm needs one $D_3(n/2)$ and three $D_\oplus$. Therefore, recursive use of this algorithm has the following complexities:

$$\begin{cases} M_{3,\otimes}(n) \leq 3M_{3,\otimes}(n/2), \ M_\otimes = 1, \\ M_{3,\oplus}(n) \leq 3M_{3,\oplus}(n/2) + 4n - 4, \ M_\oplus(1) = 0 \\ M_3(n) \quad \leq 3M_3(n/2) + 4n - 4, \ M(1) = 1 \\ D_3(n) \quad \leq D_3(n/2) + 3D_\oplus, D_3(1) = D_\otimes. \end{cases} \tag{2}$$

Using Theorem 1 we obtain the following bounds:

$$\begin{cases} M_{3,\otimes}(n) \leq n^{\log_2 3}, \\ M_{3,\oplus}(n) \leq 6n^{\log_2 3} - 8n + 2, \\ M_3(n) \quad \leq 7n^{\log_2 3} - 8n + 2, \\ D_3(n) \quad \leq 3\log_2 n D_\oplus + D_\otimes. \end{cases} \tag{3}$$

3.2 Improved Karatsuba 2-way algorithm

By using the algorithm given in [3], one can improve the reconstruction part of the algorithm given in (1) as follows:

$$\begin{cases} P_0 = A_0B_1, \;\; P_1 = (A_0 + A_1)(B_0 + B_1), \;\; P_2 = A_2B_2, \\ C = (X^{n/2} - 1)(X^{n/2}P_2 - P_0) + P_1X^n. \end{cases} \tag{4}$$

The algorithm given in (4) requires three multiplications of two degree $n - 1$ polynomials plus $7n/2 - 3$ additions. On the other hand, the delay complexity of the algorithm needs one $D_3(n/2)$ and three $D_\oplus$. Therefore, using this algorithm recursively has the following complexities:

$$\begin{cases} M_{3,\otimes}(n) \leq 3M_{3,\otimes}(n/2), \; M_\otimes = 1, \\ M_{3,\oplus}(n) \leq 3M_{3,\oplus}(n/2) + 7n - 3, \; M_\oplus(1) = 0, \\ M_3(n) \quad \leq 3M_3(n/2) + 7n - 3, \; M(n) = 1, \\ D_3(n) \quad \leq D_3(n/2) + 3D_\oplus, D_3(1) = D_\otimes. \end{cases} \tag{5}$$

We get the following bounds by using Theorem 1:

$$\begin{cases} M_{3,\otimes}(n) \leq n^{\log_2 3}, \\ M_{3,\oplus}(n) \leq 5.5n^{\log_2 3} - 7n + 1.5, \\ M_3(n) \quad \leq 6.5n^{\log_2 3} - 7n + 1.5, \\ D_3(n) \quad \leq 3\log_2 nD_\oplus + D_\otimes. \end{cases} \tag{6}$$

3.3 Karatsuba like 3-way algorithm

Let $A = \sum_{i=0}^{n-1} a_iX^i$ and $B = \sum_{i=0}^{n-1} b_iX^i$. We can divide A and B into three parts as follows: $A(X) = A_0 + A_1X^{n/3} + A_2X^{2n/3}$ and $B(X) = B_0 + B_1X^{n/3} + B_2X^{2n/3}$ where A_i and B_i are polynomials of degree less than $n/3$. Let $C = \sum_{i=0}^{4} C_iX^{in/3}$ be the product of A and B. A Karatsuba like 3-way algorithm that can be obtained by using Chinese remainder theorem [18] is the following:

$$\begin{cases} P_0 = A_0B_0, \;\; P_1 = A_1B_1, \;\; P_2 = A_2B_2, \;\; P_3 = (A_0 + A_1)(B_0 + B_1), \\ P_4 = (A_0 + A_2)(B_0 + B_2) \;\; P_5 = (A_1 + A_2)(B_1 + B_2). \\ C = P_0 + (P_3 - P_0 - P_1)X^{n/3} + (P_4 + P_1 - P_0 - P_2)X^{2n/3} + \\ (P_5 - P_1 - P_2)X^{3n/3} + P_2X^{4n/3}. \end{cases} \tag{7}$$

The algorithm given in (7) requires six multiplications of two degree $n/3 - 1$ polynomials plus $2n$ additions for P_i's; $14n/3 - 7$ additions for coefficients of C and $4n/3 - 4$ additions for overlaps. On the other hand, the delay complexity of the algorithm is one $D_3(n/2)$ and four $D_\oplus$. Therefore, the recursive use of this algorithm has the following complexities:

$$\begin{cases} M_{3,\otimes}(n) \leq 6M_{3,\otimes}(n/3), \; M_\otimes(1) = 1, \\ M_{3,\oplus}(n) \leq 6M_{3,\oplus}(n/3) + 8n - 11, \; M_\oplus(1) = 0, \\ M_3(n) \quad \leq 6M_3(n/3) + 8n - 11, \; M(1) = 1, \\ D_3(n) \quad \leq D_3(n/3) + 4D_\oplus, D_3(1) = D_\otimes. \end{cases} \tag{8}$$

We obtain the following bounds by using Theorem 1:

$$\begin{cases} M_{3,\otimes}(n) \leq n^{\log_3 6}, \\ M_{3,\oplus}(n) \leq 5.8n^{\log_3 6} - 8n + 2.2, \\ M_3(n) \quad \leq 6.8n^{\log_3 6} - 8n + 2.2, \\ D_3(n) \quad \leq 4\log_3 nD_\oplus + D_\otimes. \end{cases} \tag{9}$$

3.4 Improved Karatsuba like 3-way algorithm

In this section we redesign the reconstruction part of the algorithm in (7) by using the similar technique in [19]. Note that the degree of products P_i's for $0 \leq i \leq 4$ is $2n/3 - 2$. We divide each P_i into two parts as $P_i = P_{iL} + x^{n/3}P_{iH}$ where P_{iL} is a degree $n/3 - 1$ polynomial and P_{iH} is a degree $n/3-2$ polynomial. After substituting those representation of products into the reconstruction part of (7) we get the following:

$$\begin{aligned} C = P_{0L} + X^{n/3}(P_{0H} - P_{0L} - P_{1L} + P_{3L}) + X^{2n/3}(-P_{0L} - P_{0H} + P_{1L} - P_{1H} - P_{2L} + P_{3H} + P_{4L}) + \\ X^{3n/3}(-P_{0H} + P_{1H} - P_{2L} - P_{2H} + P_{4H} + P_{5L} - P_{1L}) + X^{4n/3}(-P_{1H} + P_{2L} - P_{2H} + P_{5H}) + X^{5n/3}P_{2H}. \end{aligned} \tag{10}$$

Note that there are no overlaps in (10). Therefore we just compute the cost of the coefficients. To improve the algorithm one can observe some common terms $R_1 = P_{0H} - P_{1L}$ and $R_2 = P_{1H} - P_{2L}$ in (10). Then we have,

$$\begin{aligned} C = P_{0L} + X^{n/3}(R_1 - P_{0L} + P_{3L}) + X^{2n/3}(-R_1 - P_{0L} - P_{1H} - P_{2L} + P_{3H} + P_{4L}) + \\ X^{3n/3}(R_2 - P_{0H} - P_{1L} - P_{2H} + P_{4H} + P_{5L}) + X^{4n/3}(-R_2 - P_{2H} + P_{5H}) + X^{5n/3}P_{2H}. \end{aligned} \tag{11}$$

The number of additions required in (11) is computed as follows: Computing $(A_0 + A_1), (B_0 + B_1), (A_0 + A_2), (B_0 + B_2), (A_1 + A_2), (B_1 + B_2)$ require $2n$ additions. Computing each of R_1 and R_2 requires $n/3 - 1$ additions. On the other hand, we need $2n/3$ additions for $(R_1 - P_{0L} + P_{3L})$, $5n/3 - 2$ additions for $-R_1 - P_{0L} - P_{1H} - P_{2L} + P_{3H} + P_{4L}$, $5n/3 - 3$ additions for $R_2 - P_{0H} - P_{1L} - P_{2H} + P_{4H} + P_{5L}$ and $2n/3 - 2$ additions for $-R_2 - P_{2H} + P_{5H}$. The delay complexity of the algorithm is the same with the previous one. Therefore, the recursive use of this algorithm has the following complexities:

$$\begin{cases} M_{3,\otimes}(n) \leq 6M_{3,\otimes}(n/3),\ M_\otimes = 1, \\ M_{3,\oplus}(n) \leq 6M_{3,\oplus}(n/3) + 22n/3 - 9,\ M_\oplus(1) = 0, \\ M_3(n) \quad \leq 6M_3(n/3) + 21n/3 - 9,\ M(n) = 1, \\ D_3(n) \quad \leq D_3(n/3) + 4D_\oplus, D(1) = D_\otimes. \end{cases} \tag{12}$$

The solutions can be obtained by using Theorem 1 as follows:

$$\begin{cases} M_{3,\otimes}(n) \leq n^{\log_3 6}, \\ M_{3,\oplus}(n) \leq 5.53n^{\log_3 6} - 7.33n + 1.8, \\ M_3(n) \quad \leq 6.53n^{\log_3 6} - 7.33n + 1.8, \\ D_3(n) \quad \leq 4\log_3 nD_\oplus + D_\otimes. \end{cases} \tag{13}$$

4 Proposed 3-way algorithm with five multiplications

In this section, we give a 3-way algorithm for multiplying polynomials of degree $(3n-1)$ over $\mathbb{F}_3$ with five multiplications using interpolation in $\mathbb{F}_9$. Let $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$. We can divide A and B into three parts as follows: $A(X) = A_0 + A_1 X^{n/3} + A_2 X^{2n/3}$ and $B(X) = B_0 + B_1 X^{n/3} + B_2 X^{2n/3}$ where A_i and B_i are polynomials of degree less than $n/3$. Let $C = \sum_{i=0}^{4} C_i X^{in/3}$ be the product of A and B. Recall that $\mathbb{F}_9 = \mathbb{F}_3[X]/ < X^2+1 >$ and ω is a root of $X^2+1=0$ in $\mathbb{F}_9$.

In order to obtain an algorithm for the product $C = AB$ with five multiplications, we will use the interpolation method which yields Toom-Cook like formulas [18], [11], [16]. Since there are not enough points in $\mathbb{F}_3$ for interpolation method, we will use an element from $\mathbb{F}_9$, i.e., we will use the points $0, 1, 2, \infty$ and w as evaluation points. Then, evaluations of those points in $AB = C$ gives us the following system of linear equations in $\mathbb{F}_9$:

Evaluation at $X = 0 \Longrightarrow P_0 = A_0 B_0 = C_0$
Evaluation at $X = 1 \Longrightarrow P_1 = (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) = C_0 + C_1 + \cdots + C_4$
Evaluation at $X = -1 \Longrightarrow P_2 = (A_0 - A_1 + A_2)(B_0 - B_1 + B_2) = C_0 - C_1 + \cdots + C_4$
Evaluation at $X = w \Longrightarrow P_3 = (A_0 + A_1 w - A_2)(B_0 + B_1 w - B_2) = C_0 + C_1 w - \cdots + C_4$
Evaluation at $X = \infty \Longrightarrow P_4 = A_2 B_2 = C_4$.

Solving this system of linear equations gives us the following algorithm for computing the product $C = AB$:

$$\begin{cases} C_0 = P_0, \\ C_1 = (P_1 - P_2) - (-P_0 + P_1 + P_2 - P_3 - P_4)w, \\ C_2 = -(P_0 + P_1 + P_2 + P_4), \\ C_3 = (P_1 - P_2) + (-P_0 + P_1 + P_2 - P_3 - P_4)w, \\ C_4 = P_4. \end{cases} \tag{14}$$

Now we will compute the cost of the recursive use of this algorithm. Assume that A and B are degree $n-1$ polynomials. Therefore, A_0, A_1, A_2, B_0, B_1 and B_2 are degree $(n/3-1)$ polynomials. The cost of operations are given in Table 1.

Remark 2. Note that for $\mathbb{F}_3$ computations, the costs of U_4 and U_5 are zero since these are related to the ω-free part of the results.

In order to compute the delay complexity for multiplication in $\mathbb{F}_9[X]$, we draw multi-evaluation and reconstruction data flow in Figure 1. As it can be seen from the figure, the evaluation requires two additions. The critical path starts from A_0 and ends at R_2. On the other hand, the critical path of reconstruction needs four additions that starts from A_1 and goes through U_2, U_4, U_5 and ends at C_1. Note that multiplication by ω is free of cost. Therefore, we don't count it in the complexity analysis. Finally, we need one addition in the final overlap and therefore we obtain the complexities in (15). As a result, we obtain the following complexities:

Table 1. Cost of multi-evaluation and reconstruction for the new three-way split formulas

Computations	Cost in $\mathbb{F}_3$ for multiplication in $\mathbb{F}_9[X]$	Cost in $\mathbb{F}_3$ for multiplication in $\mathbb{F}_3[X]$
$R_1 = A_0 + A_2$, $R'_1 = B_0 + B_2$	$4n/3$	$2n/3$
$R_2 = R_1 + A_1$, $R'_2 = R'_1 + B_1$	$4n/3$	$2n/3$
$R_3 = R_1 - A_1$, $R'_3 = R'_1 - B_1$	$4n/3$	$2n/3$
$R_4 = wA_1$, $R'_4 = wB_1$	0	0
$R_5 = A_0 - A_2$, $R'_5 = B_0 - B_2$	$4n/3$	$2n/3$
$R_6 = R_4 + R_5$, $R'_6 = R'_4 + R'_5$	$4n/3$	0
$P_0 = A_0B_0$	$M_9(n/3)$	$M_3(n/3)$
$P_1 = R_2R'_2$	$M_9(n/3)$	$M_3(n/3)$
$P_2 = R_3R'_3$	$M_9(n/3)$	$M_3(n/3)$
$P_3 = R_6R'_6$	$M_9(n/3)$	$M_3(n/3)$
$P_4 = A_2B_2$	$M_9(n/3)$	$M_3(n/3)$
$U_1 = P_1 - P_2$	$4n/3 - 2$	$2n/3 - 1$
$U_2 = P_1 + P_2$	$4n/3 - 2$	$2n/3 - 1$
$U_3 = P_0 + P_4$	$4n/3 - 2$	$2n/3 - 1$
$C_2 = -(U_2 + U_3)$	$4n/3 - 2$	$2n/3 - 1$
$U_4 = U_2 - U_3$	$4n/3 - 2$	0
$U_5 = U_4 - P_3$	$4n/3 - 2$	0
$U_6 = wU_5$	0	0
$C_3 = U_1 + U_6$	$4n/3 - 2$	$2n/3 - 1$
$C_1 = U_1 - U_6$	$4n/3 - 2$	$2n/3 - 1$
$C = P_0 + C_1X^{n/3} + C_2X^{2n/3} + C_3X^{3n/3} + C_4X^{4n/3}$	$8n/3 - 8$	$4n/3 - 4$
Total	$5M_9(n/3) + 60n/3 - 24$	$4M_3(n) + M_9(n) + 24n - 10$

$$\begin{cases} M_{9,\otimes}(n) \le 5M_{9,\otimes}(n/3),\ M_{9,\otimes}(1) = 4, \\ M_{9,\oplus}(n) \le 5M_{9,\oplus}(n/3) + 20n - 24,\ M_{9,\oplus}(1) = 2, \\ M_9(n) \le 5M_9(n/3) + 20n - 24,\ M(1) = 6, \\ D_9(n) \le D_9(n/3) + 7D_\oplus, D_9(1) = D_\oplus + D_\otimes. \end{cases} \tag{15}$$

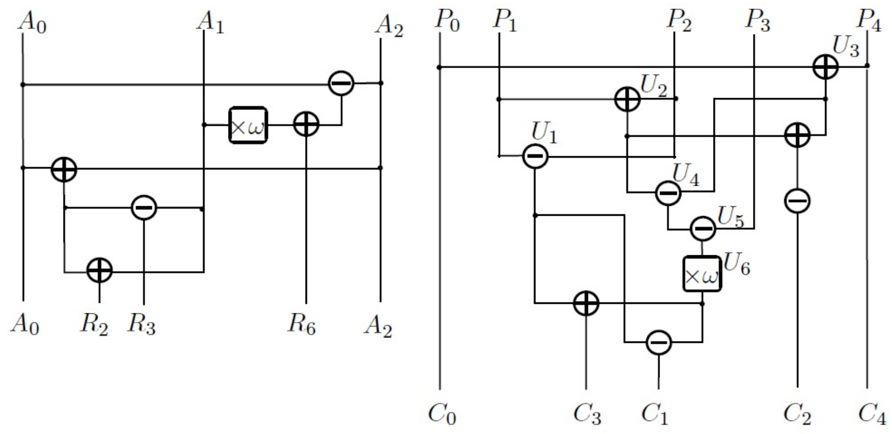
We get the following bounds by using Theorem 1:

$$\begin{cases} M_{9,\otimes}(n) \le 4n^{\log_3 5}, \\ M_{9,\oplus}(n) \le 26^{\log_3 5} - 30n + 6, \\ M_9(n) \quad \le M_9(n) \le 30n^{\log_3 5} - 30n + 6, \\ D_9(n) \quad \le (7\log_3 n + 1)D_\oplus + D_\otimes. \end{cases} \tag{16}$$

Now we obtain the following complexity for $M_3(n)$ by substituting the result from (15) into $M_3(n)$ and using Theorem 1:

$$\begin{cases} M_{3,\otimes}(n) \le 4M_\otimes(n/3) + M_{9,\otimes}(n/3),\ M_\otimes(1) = 1, \\ M_{3,\oplus}(n) \le 4M_{3,\oplus}(n/3) + M_{9,\oplus}(n/3) + 8n - 10,\ M_\oplus(1) = 0 \\ M_3(n) \quad \le 4M_3(n/3) + M_9(n/3) + 8n - 10,\ M(1) = 6 \\ D_3(n) \quad \le D_9(n/3) + 7D_\oplus. \end{cases} \tag{17}$$

Fig. 1. Multi-evaluation (left) and reconstruction (right) data flow for multiplication in $\mathbb{F}_9[X]$



$$\begin{cases} M_{3,\otimes}(n) \leq 4n^{\log_3 5} - 3n^{\log_3 4}, \\ M_{3,\oplus}(n) \leq 26^{\log_3 5} - 33.33n^{\log_3 4} + 6n + 1.33, \\ M_3(n) \quad \leq 30^{\log_3 5} - 36.33n^{\log_3 4} + 6n + 1.33, \\ D_3(n) \quad\; \leq (7\log_3 n + 1)D_\oplus + D_\otimes. \end{cases} \tag{18}$$

We summarize the complexities of the algorithms in Table 2.

Table 2. Complexities of the different approaches for multiplication over $\mathbb{F}_3$

Algorithm	$M_{3,\oplus}(n)$	$M_{3,\otimes}(n)$	Delay
2-way (Section 3.1)	$6n^{\log_2 3} - 8n + 2$	$n^{\log_2 3}$	$3\log_2 nD_\oplus + D_\otimes$
Improved 2-way (Section 3.2)	$5.5n^{\log_2 3} - 7n + 1.5$	$n^{\log_2 3}$	$3\log_2 nD_\oplus + D_\otimes$
3-way (Section 3.3)	$5.8n^{\log_3 6} - 8n + 2.2$	$n^{\log_3 6}$	$4\log_3 nD_\oplus + D_\otimes$
Improved 3-way (Section 3.4)	$5.53n^{\log_3 6} - 7.33n + 1.8$	$n^{\log_3 6}$	$4\log_3 nD_\oplus + D_\otimes$
New 3-way (Section 4)	$26n^{\log_3 5} - 33.33n^{log_3 4}$ $+6n + 1.33$	$4n^{\log_3 5} - 3n^{log_3 4}$	$(7\log_3 n + 1)D_\oplus + D_\otimes$

5 Multiplication in $\mathbb{F}_{3^{6n}}$

In this section, we show that the proposed algorithm of Section 4 for multiplication in $\mathbb{F}_9[X]$ yields considerable improvements for the multiplication in $\mathbb{F}_{3^{6n}}$. The main idea comes from representing $\mathbb{F}_{3^{6n}}$ as $\mathbb{F}_{9^{3n}}$ and then using the proposed algorithm recursively for multiplication in $\mathbb{F}_9[X]$. Note that classical way for performing multiplication in $\mathbb{F}_{3^{6n}}$ is to obtain a formula that uses multiplications in $\mathbb{F}_{3^m}$ and then using classical algorithms for multiplication in $\mathbb{F}_{3^m}$. To our knowledge, the best arithmetic complexity with this approach can be obtained using the algorithms in [16] and [12] with 15 multiplications in $\mathbb{F}_{3^n}$. We propose a different approach. Instead of using algorithms that compute multiplication in $\mathbb{F}_{3^{6n}}$ in terms of $\mathbb{F}_{3^n}$, we represent $\mathbb{F}_{3^{6n}}$ as $\mathbb{F}_{9^{3n}}$ and then recursively applying the proposed algorithm of Section 4 for multiplying $3n$-term polynomials over $\mathbb{F}_9$.

Note that the multiplications in $\mathbb{F}_{3^{6n}}$ required for pairing computations includes both sparse and dense multiplications. For example, the number of sparse multiplications required in μ_T pairing computation is $(m+1)/2$ and there is only one dense multiplication needed in $\mathbb{F}_{3^{6n}}$ that requires in the exponentiation step of μ_T pairing computation. In [4] and [17], it was shown that the sparse multiplication in the μ_T pairing computation can be performed using 13 multiplications and 50 additions in $\mathbb{F}_{3^n}$. Moreover, it can be easily observed that the cost can be reduced to 12 multiplications and 51 additions in $\mathbb{F}_{3^m}$ by using the algorithm in [16]. After that classical multiplication methods are used for multiplication in $\mathbb{F}_{3^m}$.

On the other hand, for dense multiplication the first used method was the tower representation of $\mathbb{F}_{3^{6n}}$ in which elements are represented by three-term polynomials with coefficients from $\mathbb{F}_{3^{2m}}$ which are also represented by two term polynomials with coefficients from $\mathbb{F}_{3^m}$. Since the multiplication of three term polynomials requires six multiplications and the multiplication of two-term polynomials takes two multiplications, this method requires 18 multiplication in $\mathbb{F}_{3^m}$. On the other hand, in [16] and [12] it was shown that this kind of multiplications can be performed with 15 multiplications in $\mathbb{F}_{3^m}$ by using discrete fourier transform and Chinese remainder theorem, respectively.

Below, we propose our new approach for the sparse and the dense multiplications. This new approach requires $5M_9(n)$ for dense polynomial multiplications and $4M_9(n)$ for sparse polynomial multiplications. When we compare those arithmetic costs with $15M_3(n)$ and $12M_3(n)$, we obtain improvements between 14 and 22%.

5.1 Dense multiplication in $\mathbb{F}_{3^{6n}}$

By using the representation $\mathbb{F}_{3^{6n}} \cong \mathbb{F}_{9^{3n}}$ and recursively applying the algorithm given in (14) for multiplying polynomials in $\mathbb{F}_9[X]$, we observe that

$$M_3(6n) \leq M_9(3n) \leq 5M_9(n) + 20n - 24.$$

Note that for practical values of n the linear term $20n - 24$ is negligible compared to $5M_9(n)$. Similarly, recall that the best known algorithm for multiplication in $\mathbb{F}_{3^{6n}}$ requires $15M_3(n)$ plus some negligibly small number of additions. Asymptotically, we have the following: Using complexity with improved Karatsuba 2-way, $15M_3(n)$ requires $97.5n^{1.58} + O(n)$ operations and using improved Karatsuba 3-way, $15M_3(n)$ needs $82.15n^{1.63} + O(n)$ operations. Moreover, using the proposed method for multiplication in $\mathbb{F}_9[X]$, $5M_9(n)$ needs $150n^{1.46} + O(n)$ and it can be concluded that proposed method yields better results.

On the other hand, the best values of $15M_3(n)$ and $5M_9(n)$ for some values of n used in cryptography are given in Section 5.3.

5.2 Sparse multiplication in $\mathbb{F}_{3^{6n}}$

There are $(m+1)/2$ sparse multiplications needed in $\mathbb{F}_{3^{6n}} \cong \mathbb{F}_{9^{3n}}$ for the μ_T pairing computation. Assume that $A, B \in \mathbb{F}_{9^{3n}}$ are to be multiplied in the computation of the μ_T pairing computation. Here, the first operand A is a three-term dense polynomial over $\mathbb{F}_{9^n}$; however B is sparse polynomial over $\mathbb{F}_{9^n}$. More specifically, we can assume that the representation of A and B are the following form:

$$A = A_0 + A_1X + A_2X^2, \;\; B = B_0 + B_1X + B_2X^2$$

where $A_i \in \mathbb{F}_{9^n}$ for $i = 0, 1, 2$ and $B_0 \in \mathbb{F}_{9^n}$, $B_1 \in \mathbb{F}_{3^m}$ and $B_2 = -1 \in \mathbb{F}_3$. For more detail of the representations we refer the reader to [4], [17], [9]. We obtain the following algorithm by using the algorithm given in (14):

$$\begin{cases} P_0 = A_0B_0, \\ P_1 = (A_0 + A_1 + A_2)(B_0 + B_1 - 1), \\ P_2 = (A_0 - A_1 + A_2)(B_0 - B_1 - 1), \\ P_3 = (A_0 + A_1\omega - A_2)(B_0 + B_1\omega + 1), \\ P_4 = -A_2, \\ C_0 = P_0, \\ C_1 = (P_1 - P_2) + (P_0 - P_1 - P_2 + P_3 + P_4)\omega, \\ C_2 = -(P_0 + P_1 + P_2 + P_4), \\ C_3 = (P_1 - P_2) + (-P_0 + P_1 + P_2 - P_3 - P_4)\omega, \\ C_4 = -A_2. \end{cases} \tag{19}$$

This algorithms requires $4M_9(n)+O(n)$. Note that the classical method requires $12M_3(n)+O(n)$. Since the proposed algorithm has coefficient which is three times less than the known algorithm, we have the similar asymptotical results as in Section 5.1. The best values of $15M_3(n)$ and $5M_9(n)$ for some values of n used in cryptography are given in below.

5.3 Improvements in arithmetic complexity

In this section we compute the arithmetic cost of multiplication in $\mathbb{F}_{3^{6n}}$ for $n = 167$, 193, 239, 317, 353, 509 which are used in pairing based cryptography. First, note that we use 2-way or 3-way splits. If n is not divisible by two or three, we pad one or two zeros to the polynomial so that the sizes become divisible by three. This effects the complexity results negligibly. Second, note that using the same algorithm in all recursion until the size becomes one does not give the best results. Using the schoolbook method after the size becomes enough small yields better result. Remember that the schoolbook method require n^2 multiplications and $(n-1)^2$ additions in order to multiply two $(n-1)$ degree polynomials or it gives

$$M_3(n+1) \leq M_3(n) + 4n. \tag{20}$$

When we refer to the schoolbook method it should be understood that we are computing $M_3(n+1)$ in terms of $M_3(n)$.

In order to show the effect of this approach, we can consider, for example, $M_3(8)$. Using the improved Karatsuba method in each recursion gives

$$M_3(8) = 3M_3(4) + 25 = 9M_3(2) + 58 = 27M_3(1) + 94 = 123.$$

On the other hand, using the schoolbook after $n = 4$ gives

$$M_3(8) = 3M_3(4) + 25 = 3.25 + 25 = 100.$$

We will use the same strategy for the multiplication in $\mathbb{F}_9[X]$. We observe that using school method and improved Karatsuba 2-way method for multiplication in $\mathbb{F}_9[X]$ brings better results for multiplication in $\mathbb{F}_9[X]$ for small values of n. Remember from Section 2 that we have $M_9(1) = 6$. It can be easily derived that improved Karatsuba 2-way method for multiplication in $\mathbb{F}_9[X]$ gives

$$M_9(n) \leq 3M_9(n/2) + 7n - 6, \tag{21}$$

and school method in $\mathbb{F}_9[X]$ gives

$$M_9(n+1) \leq M_9(n) + 16n + 4. \tag{22}$$

Moreover, we use another algorithm and proceed as follows: We split the degree $(n-1)$ polynomials that will be multiplied into two parts by extracting ω part and ω-free part, i.e., we write $A, B \in \mathbb{F}_9[X]$ as $A = A_0 + A_1\omega$ and $B = B_0 + B_1\omega$ where $A_0, A_1, B_0, B_1 \in \mathbb{F}_3[X]$ of degree $(n-1)$. Then

$$AB = (A_0+A_1\omega)(B_0+B_1\omega) = A_0B_0 - A_1B_1 + ((A_0+A_1)(B_0+B_1) - A_0B_0 - A_1B_1)\omega, \tag{23}$$

which requires $3M_3(n) + 8n - 3$ and gives $M_9(3) \leq 60$.

Now we are ready to compute the arithmetic cost of multiplication in $\mathbb{F}_{3^{6n}}$ for $n = 167$, 193, 249, 317, 353, 509. We use the following abbreviations for the algorithm names:

- KA for improved Karatsuba 2-way in $\mathbb{F}_3[X]$ given in Section 3.2
- SB for the schoolbook method in $\mathbb{F}_3[X]$
- KA_9 for improved Karatsuba 2-way in $\mathbb{F}_9[X]$ given by (21)
- $A1_9$ for the new 3-way algorithm for multiplication in $\mathbb{F}_9[X]$ in Section 4
- $A2_9$ for multiplication in $\mathbb{F}_9[X]$ given by (23)

Note that when we use an algorithm A recursively ℓ times , we write $(A)^\ell$. The results in Table 3 and 4 show that using 3-way algorithm with five multiplications in $\mathbb{F}_9$ improves the old ones significantly.

6 Conclusion

In this paper, we have proposed improved algorithms for multiplication $\mathbb{F}_{3^n}$. First, we presented some improvements on the classical Karatsuba algorithm over binary fields that

Table 3. Complexities for polynomial multiplication over $\mathbb{F}_3$ and $\mathbb{F}_9$

n	Multiplication in $\mathbb{F}_3[X]$		Multiplication in $\mathbb{F}_9[X]$		Ratio
	Algorithms used	Total cost	Algorithms used	Total cost	$M_9(n)/M_3(n)$
167	$(K2')^6, (SB)^3$	21762	$(A1_9)^2, KA_9, A2_9, KA, (SB)^5$	52916	2.43
193	$(K2')^5$, $(SB)^7$	30001	$(A1_9)^2, A2_9, (KA)^3, (SB)^3$	67481	2.25
239	$(K2')^6, (SB)^4$	35298	$(A1_9)^4, A2_9, (SB)^3$	82636	2.34
317	$(K2')^6, (SB)^5$	52065	$(A1_9)^4, (KA_9)^2$	123916	2.38
353	$(K2')^7, (SB)^3$	67761	$(A1_9)^4, A2_9, (SB)^5$	173836	2.57
509	$(K2')^7, (SB)^4$	109041	$(A1_9)^3, KA_9, A2_9, KA, (SB)^5$	275056	2.52

Table 4. Cost of multiplication in $\mathbb{F}_{3^{6n}}$

Type of polynomial	n	Old (recursion in $\mathbb{F}_3[X]$)	New (recursion in $\mathbb{F}_9[X]$)	Improvement(%)
Dense	167	326430	264580	18.95
	193	450015	337405	25.02
	239	529470	413180	21.96
	317	780975	619580	20.67
	353	1016415	869180	14.49
	509	1635615	1375280	15.92
Sparse	167	261144	211664	18.95
	193	360012	269924	25.02
	239	423576	330544	21.96
	317	624780	495664	20.67
	353	813132	695344	14.49
	509	1308492	1100224	15.92

can also be used in characteristic three fields and we have given the cost of the improved Karatsuba 2-way and 3-way algorithms. Then, we have derived a 3-way polynomial multiplication algorithm with five 1/3 size multiplications using interpolation in $\mathbb{F}_9$. Moreover, the arithmetic and delay complexities of the recursive use of this algorithm have been computed. Finally, it has been showed that recursively using the proposed 3-way algorithm in $\mathbb{F}_9$ yields about 14% to 22% improvements for the multiplication in $\mathbb{F}_{3^{6n}}$ for $n = 167$, 193, 249, 317, 353, 509 which are widely used in pairing based cryptography.

References

1. Omran Ahmadi, Darrel Hankerson, and Alfred Menezes. Formulas for cube roots in $\mathrm{F}_{3}\mathrm{m}$. *Discrete Applied Mathematics*, 155(3), 2007.

2. Omran Ahmadi, Darrel Hankerson, and Alfred Menezes. Software implementation of arithmetic in F_{3^m}. In *WAIFI*, pages 85–102, 2007.
3. D.J. Bernstein. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336, 2009.
4. Guido Marco Bertoni, Luca Breveglieri, Pasqualina Fragneto, and Gerardo Pelosi. Parallel hardware architectures for the cryptographic tate pairing. *I. J. Network Security*, 7(1):31–37, 2008.
5. Jean-Luc Beuchat, Nicolas Brisebarre, Jérémie Detrey, Eiji Okamoto, and Francisco Rodríguez-Henríquez. A comparison between hardware accelerators for the modified tate pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$. In *Pairing*, pages 297–315, 2008.
6. Jean-Luc Beuchat, Nicolas Brisebarre, Jérémie Detrey, Eiji Okamoto, Masaaki Shirase, and Tsuyoshi Takagi. Algorithms and arithmetic operators for computing the η_T pairing in characteristic three. *IEEE Trans. Computers*, 57(11):1454–1468, 2008.
7. Jean-Luc Beuchat, Jérémie Detrey, Nicolas Estibals, Eiji Okamoto, and Francisco Rodríguez-Henríquez. Hardware accelerator for the tate pairing in characteristic three based on Karatsuba-Ofman multipliers. In *CHES*, pages 225–239, 2009.
8. Jean-Luc Beuchat, Jérémie Detrey, Nicolas Estibals, Eiji Okamoto, and Francisco Rodríguez-Henríquez. Fast architectures for the η_t pairing over small-characteristic supersingular elliptic curves. *IEEE Trans. Computers*, 60(2):266–281, 2011.
9. Jean-Luc Beuchat, Hiroshi Doi, Kaoru Fujita, Atsuo Inomata, Piseth Ith, Akira Kanaoka, Masayoshi Katouno, Masahiro Mambo, Eiji Okamoto, Takeshi Okamoto, Takaaki Shiga, Masaaki Shirase, Ryuji Soga, Tsuyoshi Takagi, Ananda Vithanage, and Hiroyasu Yamamoto. FPGA and ASIC implementations of the η_T pairing in characteristic three. *Computers & Electrical Engineering*, 36(1):73–87, 2010.
10. Jean-Luc Beuchat, Emmanuel López-Trejo, Luis Martínez-Ramos, Shigeo Mitsunari, and Francisco Rodríguez-Henríquez. Multi-core implementation of the tate pairing over supersingular elliptic curves. In *CANS*, pages 413–432, 2009.
11. M. Cenk, C. Negre, and M. Anwar Hasan. Improved three-way split formulas for binary polynomial and toeplitz matrix vector products. *IEEE Transactions on Computers*. to appear.
12. Murat Cenk and Ferruh Özbudak. Efficient multiplication in $\mathbb{F}_{3^{\ell m}}, m \geq 1$ and $5 \leq \ell \leq 18$. In *AFRICACRYPT*, pages 406–414, 2008.
13. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.
14. Nicolas Estibals. Compact hardware for computing the tate pairing over 128-bit-security supersingular curves. In *Pairing*, pages 397–416, 2010.
15. Reza Rezaeian Farashahi, Hongfeng Wu, and Changan Zhao. Efficient arithmetic on hessian curves over fields of characteristic three. *IACR Cryptology ePrint Archive*, 2012, 2012.
16. Elisa Gorla, Christoph Puttmann, and Jamshid Shokrollahi. Explicit formulas for efficient multiplication in $F_{3^{6m}}$. In *Selected Areas in Cryptography*, pages 173–183, 2007.
17. R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS J. Comput. Math.*, 9:64–85, 2006.
18. S. Winograd. *Arithmetic Complexity of Computations.* Society For Industrial & Applied Mathematics, U.S., 1980.
19. G. Zhou and H. Michalik. Comments on "a new architecture for a parallel finite field multiplier with low complexity based on composite field". *IEEE Trans. Computers*, 59(7):1007–1008, 2010.