J Sign Process Syst (2018) 90:285–294
DOI 10.1007/s11265-017-1234-x

New Efficient Algorithms for Multiplication Over Fields of Characteristic Three

Murat Cenk[1] · Farhad Haghighi Zadeh[2] · M. Anwar Hasan[3]

Received: 13 May 2014 / Revised: 24 November 2016 / Accepted: 20 February 2017 /Published online: 6 March 2017

Abstract In this paper, we first present an enhancement of the well-known Karatsuba 2-way and 3-way algorithms for characteristic three fields, denoted by $\mathbb{F}_{3^n}$ where $n \geq 1$. We then derive a 3-way polynomial multiplication algorithm with five $1/3$ sized multiplications that use interpolation in $\mathbb{F}_9$. Following the computation of the arithmetic and delay complexity of the proposed algorithm, we provide the results of our hardware implementation of polynomial multiplications over $\mathbb{F}_3$ and $\mathbb{F}_9$. The final proposal is a new 3-way polynomial multiplication algorithm over $\mathbb{F}_3$ that uses three polynomial multiplications of 1/3 of the original size over $\mathbb{F}_3$ and one polynomial multiplication of 1/3 of the original size over $\mathbb{F}_9$. We show that this algorithm represents about 15% reduction of the complexity over previous algorithms for the polynomial multiplications whose sizes are of practical interest.

Keywords Polynomial multiplication · Elliptic curves · Characteristic three fields

✉ Murat Cenk
mcenk@metu.edu.tr

Farhad Haghighi Zadeh
fhaghig@synopsys.com

M. Anwar Hasan
ahasan@uwaterloo.ca

[1] Institute of Applied Mathematics at Middle East Technical University, Ankara, Turkey

[2] Synopsys Inc., Toronto, Ontario, Canada

[3] Department of Electrical, Computer Engineering at University of Waterloo, Waterloo, Ontario, Canada

1 Introduction

Multiplication in characteristic three fields, denoted by $\mathbb{F}_{3^n}$, where $n \geq 1$, is employed in curve-based cryptography. The use of these fields in elliptic curve cryptography has been discussed in [11, 14, 16, 17, 19, 21]. Examples of work related to efficient arithmetic in characteristic three fields can be found in [1–3, 20]. A common method 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 up to $(n-1)$ over $\mathbb{F}_3$, the finite field with three elements. To perform multiplication in $\mathbb{F}_{3^n}$, the polynomials are first multiplied, and the result is then reduced modulo an irreducible polynomial of degree n over $\mathbb{F}_3$. The arithmetic cost of the reduction step is linear in input size; on the other hand, the polynomial multiplication step requires a sub-quadratic complexity. The multiplication step is thus more costly than the reduction step. As a result, reducing the cost of polynomial multiplication over $\mathbb{F}_3$ directly affects the cost of multiplication in $\mathbb{F}_{3^n}$.

Our Contributions For practical cryptographic applications, polynomial multiplication schemes with low arithmetic complexity are essentially Karatsuba-like algorithms. For example, recursive uses of 2-way and 3-way algorithms have a total arithmetic complexity of $7n^{1.58} - 8n + 2$ and $6.8n^{1.63} - 8n + 2.2$, respectively. In this paper, after introducing an improved version of the 2-way and 3-way algorithms, we propose a 3-way polynomial multiplication algorithm with five multiplications using interpolation in $\mathbb{F}_9$. In contrast to the formula presented in [13], we show that the recursive use of the algorithm yields an arithmetic complexity of $15n^{1.46} - 4.85n\log_3 n - 14n$. In addition, the time delay complexity that are useful when the algorithm is mapped on to bit parallel hardware are also derived. The

final proposal is a new 3-way algorithm for multiplication of polynomials over $\mathbb{F}_3$ which uses three multiplications of polynomials of 1/3 of the input size over $\mathbb{F}_3$ and one multiplication of polynomial of $1/3$ of the input size over $\mathbb{F}_9$ with a total complexity less than $15n^{1.46}$ which is, to our knowledge, superior arithmetic complexity than that available with previously known algorithms.

Organization of the Paper The remainder of the paper is organized as follows. Notations and preliminaries used in the rest of the paper are provided in Section 2. Relevant known algorithms along with our suggestions for their improvements are presented in Section 3. The next section introduces the proposed 3-way algorithm with five 1/3 sized polynomial multiplications over $\mathbb{F}_3$ and $\mathbb{F}_9$. Section 5 reports the results of our hardware implementation. Further improvements are discussed in Section 6 and concluding remarks are made in the final section.

2 Notations and Preliminaries

This section explains the notations used in the remainder of the paper and describes a few basic algorithms. Unless otherwise stated, the fields employed in the work are assumed to be of characteristic three. The following notations are used:

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

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

A further assumption is that multiplication by -1 of a polynomial is cost free and that addition and subtraction have identical complexity. It should also be noted that the cost of the multiplication in $\mathbb{F}_9$ can be assumed to be equivalent to four multiplications and two additions in $\mathbb{F}_3$, based on the following formula:

$$(a+b\omega)(c+d\omega) = ac - bd + (bc+ad)\omega.$$

In addition, no cost is incurred for the multiplication of an element in $\mathbb{F}_9$ by ω since $(a+b\omega)\omega = -b + a\omega$.

Throughout the paper we make use of the solution to the following recurrence equation.

Lemma 1 *Let a, b, ℓ be positive integers, $n = b^\ell$, $a \neq 1$, and*

$$M(n) = aM(n/b) + cn + d + fn^\delta, \quad M(1) = e$$

(i) *If $a \neq b$ and $f = 0$ then the solution of $M(n)$ is*

$$M(n) = \left(e + \frac{bc}{a-b} + \frac{d}{a-1}\right) n^{\log_b a} - \frac{bc}{a-b}n + \frac{d}{a-1}.$$

(ii) *If $a = b$ then the solution of $M(n)$ is*

$$M(n) = \frac{fb^\delta}{b^\delta - a} n^\delta + \left(e - \frac{fb^\delta}{b^\delta - a} + \frac{d}{a-1}\right) n + cn \log_b n - \frac{d}{a-1}.$$

(iii) *If $a \neq b$ then the solution of $M(n)$ is*

$$M(n) = \frac{fb^\delta}{b^\delta - a} n^\delta + \left(e + \frac{bc}{a-b} - \frac{fb^\delta}{b^\delta - a} + \frac{d}{a-1}\right) n^{\log_b a} - \left(\frac{bc}{a-b}\right) n - \frac{d}{a-1}.$$

Proof Proofs of (i) and (ii) are in [10] and [7]. For (iii), we substitute the value of $M(n/b)$ into $M(n)$. Then, we have

$$M(n) = a(aM(n/b^2) + cn/b + d + fn^\delta/b^\delta) + cn + d + fn^\delta.$$

This equation yields

$$M(n) = a^2 M(n/b^2) + (cn + acn/b) + (d + ad) + (fn^\delta + afn^\delta/b^\delta).$$

When we substitute the value of $M(n/b^2)$ into the last equation and continue this process, we obtain

$$M(n) = a^\ell M(1) + cn(1 + a/b + \ldots + (a/b)^{\ell-1}) + d(1 + a + \ldots + a^{\ell-1}) + fn^\delta(1 + (a/b^\delta) + \ldots + (a/b^\delta)^{\ell-1}).$$

After computing the expressions in the parenthesis and using $a^\ell = a^{\log_b n} = n^{\log_b a}$, we get

$$M(n) = \frac{fb^\delta}{b^\delta - a} n^\delta + \left(e + \frac{bc}{a-b} - \frac{fb^\delta}{b^\delta - a} + \frac{d}{a-1}\right) n^{\log_b a} - \left(\frac{bc}{a-b}\right) n - \frac{d}{a-1}.$$

□

3 Known Algorithms and their Improvements

This section presents Karatsuba 2-way and 3-way algorithms for characteristic three fields. Following the work in [4] and [23] for improving the corresponding algorithms in characteristic *two*, we introduce improvements for characteristic *three*.

Remark 1 To apply recursive 2-way and 3-way algorithms, the polynomials are split in two and three parts, respectively. If n is not divisible by two or three, we pad the polynomial with one or two zeros so that the sizes become divisible by two or three. This adjustment has a negligible effect on the complexity.

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 [12, 15] is the following:

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

The algorithm given in Eq. 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 is $D_3(n/2) + 3D_\oplus$. The recursive use of this algorithm is thus associated with the following complexities:

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

Using Lemma 1, we obtain the following bounds:

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

3.2 Improved Karatsuba 2-Way Algorithm

Using the algorithm given in [4] enables us to reconstruct part of the algorithm given in Eq. 1 as follows:

$$\begin{cases} P_0 = A_0 B_0, \quad P_1 = (A_0 + A_1)(B_0 + B_1), \quad P_2 = A_2 B_2, \\ C = (X^{n/2} - 1)(X^{n/2} P_2 - P_0) + P_1 X^{n/2}. \end{cases} \tag{4}$$

The algorithm given in Eq. 4 requires three multiplications of two degree $n/2 - 1$ polynomials plus $7n/2 - 3$ additions. The delay complexity of the algorithm is $D_3(n/2) + 3D_\oplus$. Therefore, using this algorithm recursively yields the following complexity:

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

Using Lemma 1 gives the following bounds:

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

Compared to Eq. 3, using Eq. 6 results in about 7% reduction in the number of arithmetic operations. The delay complexities of Eqs. 3 and 6 are the same.

3.3 Karatsuba Like 3-Way Algorithm

As before, let $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$. This time we 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 $n/3$. To compute the product $C = AB$, a Karatsuba-like 3-way algorithm, which can be obtained using the Chinese remainder theorem [22] and from [18], can be expressed as follows:

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

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

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

Applying Lemma 1 gives the following bounds:

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

3.4 Improved Karatsuba Like 3-Way Algorithm

This section presents our redesign of the reconstruction part of the algorithm in Eq. 7 with the use of a technique similar to that reported in [6, 23]. It should be noted that the degree of P_i products for $0 \le i \le 5$ 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. Substituting those representations of the products into the reconstruction part of Eq. 7 gives the following:

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

It should be noted that Eq. 10 contains no overlaps so that we compute only the cost of the coefficients. The algorithm can be improved through the observation of some common terms $R_1 = P_{0H} - P_{1L}$ and $R_2 = P_{1H} - P_{2L}$ in Eq. 10. We then have,

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

The number of additions required in Eq. 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) and (B_1+B_2) requires $2n$ additions. Computing R_1 and R_2 requires $n/3-1$ additions each. 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 as that of the previous one. The recursive use of this algorithm results in the following complexity:

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

Applying Lemma 1 gives the following solutions:

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

The results in Eq. 13 represents a reduction of the complexity of Eq. 9 by approximately 4%. The delay complexities are the same.

4 Proposed 3-Way Algorithm with Five Multiplications

In this section, we introduce a 3-way algorithm for multiplying polynomials of degree $n-1$ 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 $\mathbb{F}_9$.

To obtain an algorithm for the product $C = AB$ with five multiplications, we use the interpolation method which yields Toom-Cook like formulas [5, 7–9, 13, 22]. Since $\mathbb{F}_3$ has insufficient points for the interpolation method, we use an element from $\mathbb{F}_9$, i.e., we use the points 0, 1, 2, ∞ and ω as evaluation points. Evaluation of $AB = C$ at those points then gives us the following system of linear equations in $\mathbb{F}_9$:

$$\begin{aligned} &\text{Evaluation at } X = 0 \Longrightarrow P_0 = A_0B_0 = C_0 \\ &\text{Evaluation at } X = 1 \Longrightarrow P_1 = (A_0 + A_1 + A_2) \\ &\quad \times(B_0 + B_1 + B_2) = C_0 + C_1 + \cdots + C_4 \\ &\text{Evaluation at } X = -1 \Longrightarrow P_2 = (A_0 - A_1 + A_2) \\ &\quad \times(B_0 - B_1 + B_2) = C_0 - C_1 + \cdots + C_4 \\ &\text{Evaluation at } X = \omega \Longrightarrow P_3 = (A_0 + A_1\omega - A_2) \\ &\quad \times(B_0 + B_1\omega - B_2) = C_0 + C_1\omega - \cdots + C_4 \\ &\text{Evaluation at } X = \infty \Longrightarrow P_4 = A_2B_2 = C_4. \end{aligned}$$

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

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

We now compute the cost of the recursive use of this algorithm. Assume that A and B are degree $n-1$ polynomials. A_0, A_1, A_2, B_0, B_1 and B_2 are therefore degree $(n/3-1)$ polynomials. The cost associated with the operations are listed in Table 1.

Remark 2 For the $\mathbb{F}_3$ computations indicated in Table 1, the cost of both U_4 and U_5 is zero since these are related to the ω-free part of the results .

To compute the delay complexity for the multiplication in $\mathbb{F}_9[X]$, we have drawn the multi-evaluation and reconstruction data flow shown in Fig. 1. As can be seen from the figure, the critical path for the evaluation requires two

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 $\mathbb{F}_3[X]$
$R_1 = A_0 + A_2$, $R'_1 = B_0 + B_2$	$4n/3$	$2n/3$
$R_2 = R_1 + A_1$, $R'_2 = R'_1 + B_1$	$4n/3$	$2n/3$
$R_3 = R_1 - A_1$, $R'_3 = R'_1 - B_1$	$4n/3$	$2n/3$
$R_4 = \omega A_1$, $R'_4 = \omega B_1$	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
$P_0 = A_0 B_0$	$M_9(n/3)$	$M_3(n/3)$
$P_1 = R_2 R'_2$	$M_9(n/3)$	$M_3(n/3)$
$P_2 = R_3 R'_3$	$M_9(n/3)$	$M_3(n/3)$
$P_3 = R_6 R'_6$	$M_9(n/3)$	$M_9(n/3)$
$P_4 = A_2 B_2$	$M_9(n/3)$	$M_3(n/3)$
$U_1 = P_1 - P_2$	$4n/3 - 2$	$2n/3 - 1$
$U_2 = P_1 + P_2$	$4n/3 - 2$	$2n/3 - 1$
$U_3 = P_0 + P_4$	$4n/3 - 2$	$2n/3 - 1$
$C_2 = -(U_2 + U_3)$	$4n/3 - 2$	$2n/3 - 1$
$U_4 = U_2 - U_3$	$4n/3 - 2$	0
$U_5 = U_4 - P_3$	$4n/3 - 2$	0
$U_6 = \omega U_5$	0	0
$C_3 = U_1 + U_6$	$4n/3 - 2$	$2n/3 - 1$
$C_1 = U_1 - U_6$	$4n/3 - 2$	$2n/3 - 1$
$C = P_0 + C_1 X^{n/3} + C_2 X^{2n/3} + C_3 X^{3n/3} + C_4 X^{4n/3}$	$8n/3 - 8$	$4n/3 - 4$
Total	$5M_9(n/3) + 60n/3 - 24$	$4M_3(n/3) + M_9(n/3) + 24n/3 - 10$

additions. It begins at A_0 and ends at R_2. On the other hand, the critical path for the reconstruction needs four additions that starts from P_1 and continues through U_2, U_4, and U_5, ending at C_1. It should be noted that multiplication by ω is cost free and not counted in the complexity analysis. The last consideration is the requirement for one addition in the final overlap, so we obtain the complexity in Eq. 15.

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

Applying Lemma 1 leads to these bounds:

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

We now obtain the following complexity for $M_3(n)$ by substituting the result from Eq. 15 into $M_3(n)$ and applying Lemma 1:

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

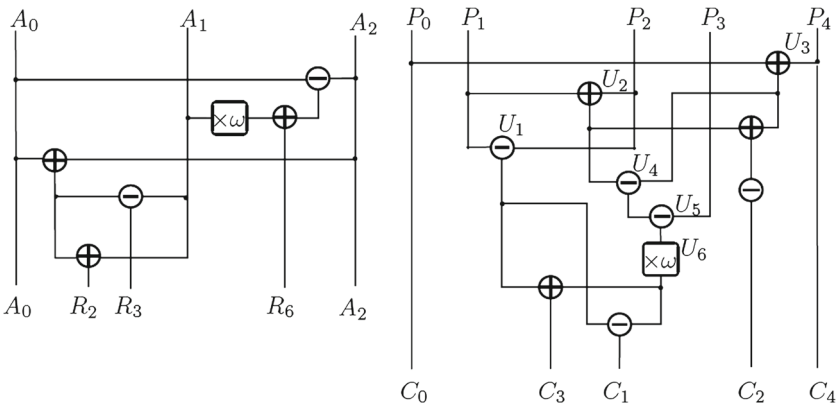

Figure 1 Multi-evaluation (*left*) and reconstruction (*right*) data flow for multiplication in $\mathbb{F}_9[X]$.

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

When we compare the total number of arithmetic operations in Eq. 18 with that in Eq. 13, which is the best known 3-way algorithm, we see that the algorithm presented in this section becomes cost effective after the size of polynomials is more than 729. The percentage of cost reduction increases significantly when the polynomial size increases. For example, the reduction is about 4% for $n = 3^7$, 14% for $n = 3^8$, 25% for $n = 3^9$, and 55% for $n = 3^{10}$. The delay complexity in Eq. 18 is about 75% higher than that in Eq. 13 and this relative difference remains the same for all practical values of n. This is because the dominant coefficients for the delay complexities in Eqs. 18 and 13 are $7\log_3 n$ and $4\log_3 n$, respectively.

5 Complexity Analysis and Implementation Results for Practical n Values

This section presents the arithmetic complexity analysis and hardware implementation of polynomial multiplications over $\mathbb{F}_3$ and $\mathbb{F}_9$ for $n = 167, 193, 239, 317$ and 353. It should first be noted that we use 2-way or 3-way splits. If n is not divisible by two or three, we pad the polynomial with one or two zeros so that the sizes become divisible by three. The adjustment has a negligible effect on the complexity. Another note is that using the same algorithm in every recursion until the size becomes unity fails to produce the best results and that employing the schoolbook method after the size becomes small enough yields a better outcome. We recall that the schoolbook method requires n^2 multiplications and $(n-1)^2$ additions in order to multiply two degree $(n-1)$ polynomials leading to

$$M_3(n+1) \le M_3(n) + 4n. \tag{19}$$

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

To demonstrate 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 method after $n = 4$ gives

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

We use the same strategy for the multiplication in $\mathbb{F}_9[X]$. It can be observed that using the schoolbook method and the improved Karatsuba 2-way method together for multiplication in $\mathbb{F}_9[X]$ yields better results for small values of n. Recalling $M_9(1) = 6$ from Section 2 leads to an easy determination that the improved Karatsuba 2-way method for multiplication in $\mathbb{F}_9[X]$ gives

$$M_9(n) \le 3M_9(n/2) + 7n - 6, \tag{20}$$

and that the schoolbook method in $\mathbb{F}_9[X]$ gives

$$M_9(n+1) \le M_9(n) + 16n + 4. \tag{21}$$

As a further refinement, we use an additional strategy 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

$$\begin{aligned} AB &= (A_0 + A_1\omega)(B_0 + B_1\omega) = A_0B_0 - A_1B_1 \\ &\quad + ((A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1)\omega, \end{aligned} \tag{22}$$

i.e., $M_9(n) \le 3M_3(n) + 8n - 3$ and $M_9(3) \le 60$.

We are now ready to compute the arithmetic cost of multiplication for $n = 167, 193, 249, 317$, and 353. We have employed the following abbreviations for the algorithm names:

- KA for the improved Karatsuba 2-way in $\mathbb{F}_3[X]$ as presented in Section 3.2
- SB for the schoolbook method in $\mathbb{F}_3[X]$
- KA_9 for the improved Karatsuba 2-way in $\mathbb{F}_9[X]$ as given by Eq. 20
- $A1_9$ for the new 3-way algorithm for multiplication in $\mathbb{F}_9[X]$, as explained in Section 4
- $A2_9$ for multiplication in $\mathbb{F}_9[X]$ as given by Eq. 22

It should be noted that when we recursively use an algorithm $A\ell$ times, we write $(A)^\ell$. The recursions listed in Tables 2 are also used in our hardware implementations.

An additional note is that the classical approach for polynomial multiplication over $\mathbb{F}_9$ is the method expressed in Eq. 22, which has

$$M_9(n)/M_3(n) \approx 3. \tag{23}$$

Our proposed algorithms reduce this ratio to 2.57 or lower for the values indicated in Table 2, representing an improvement of about 15%.

For hardware implementation using digital technologies, we represent the elements of $\mathbb{F}_3$ as (x_1, x_2) where $x_1, x_2 \in \{0, 1\}$ and the elements of $\mathbb{F}_3$, 0, 1, and 2 are represented by (0, 0), (1, 0) and (0, 1), respectively. The addition of the elements of $\mathbb{F}_3$ is performed using the method reported in

Table 2 Complexities for polynomial multiplication over $\mathbb{F}_3$ and $\mathbb{F}_9$ for special n values.

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	$(KA)^6, (SB)^3$	21762	$(A1_9)^2, KA_9, A2_9, KA, (SB)^5$	52916	2.43
193	$(KA)^5, (SB)^7$	30001	$(A1_9)^2, A2_9, (KA)^3, (SB)^3$	67481	2.25
239	$(KA)^6, (SB)^4$	35298	$(A1_9)^4, A2_9, (SB)^3$	82636	2.34
317	$(KA)^6, (SB)^5$	52065	$(A1_9)^4, (KA_9)^2$	123916	2.38
353	$(KA)^7, (SB)^3$	67761	$(A1_9)^4, A2_9, (SB)^5$	173836	2.57

[20]. Let $(x_1, x_2), (y_1, y_2), (z_1, z_2) \in \mathbb{F}_3$ such that $(x_1, x_2)+(y_1, y_2) = (z_1, z_2)$. The addition can be implemented as follows:

$$t = (x_1 \mid y_2) \hat{} (x_2 \mid y_1);$$
$$z_1 = (x_2 \mid y_2) \hat{} t;$$
$$z_2 = (x_1 \mid y_1) \hat{} t;$$

It should also be noted that the negation is as follows: $2(x_1, x_2) = -(x_1, x_2) = (x_2, x_1)$, i.e., multiplication of an element of $\mathbb{F}_3$ by -1 is essentially free of cost.

Assume that $(x_1, x_2), (y_1, y_2), (z_1, z_2) \in \mathbb{F}_3$ such that $(x_1, x_2)(y_1, y_2) = (z_1, z_2)$. We use the following method for multiplication in $\mathbb{F}_3$.

$$z_1 = (x_1 \,\&\, y_1) \mid (x_2 \,\&\, y_2);$$
$$z_2 = (x_1 \,\&\, y_2) \mid (x_2 \,\&\, y_1);$$

Now, we show the representation and operations for $\mathbb{F}_9$. Recall that we represent

$$\mathbb{F}_9 \cong \mathbb{F}_3[\omega]/(\omega^2+1) = \{0, 1, 2, \omega, \omega+1, \omega+2, 2\omega, 2\omega+1, 2\omega+2\}.$$

The elements of $\mathbb{F}_9$ are therefore represented by $a_0 + a_1\omega$, where $a_1, a_2 \in \mathbb{F}_3$. For $a, b, c, d \in \mathbb{F}_3$, the addition and multiplication in $\mathbb{F}_9$ are:

$$\begin{cases} (a+b\omega)+(c+d\omega) = (a+c)+(b+d)\omega, \\ (a+b\omega)(c+d\omega) = ac-bd+(bc+ad)\omega. \end{cases}$$

Since the elements of $\mathbb{F}_3$ are each represented as a two-tuple, we need two two-tuples for representing the elements of $\mathbb{F}_9$, i.e., for $a = a_0 + a_1\omega \in \mathbb{F}_9$, we have $a = (a_{0,1}, a_{0,2}) + (a_{1,1}, a_{1,2})\omega$ or simply $a = [(a_{0,1}, a_{0,2}), (a_{1,1}, a_{1,2})]$, where the entries are from $\{0, 1\}$.

Let $a, b \in \mathbb{F}_9$ such that $a = [(a_1, a_2), (a_3, a_4)]$ and $b = [(b_1, b_2), (b_3, b_4)]$. Then $a + b$ is obtained as follows:

$$a+b = [\underbrace{(a_1, a_2)+(b_1, b_2)}_{\text{Call } \mathbb{F}_3 \text{ addition}}, \underbrace{(a_3, a_4)+(b_3, b_4)}_{\text{Call } \mathbb{F}_3 \text{ addition}}].$$

On the other hand, multiplication of $a = [(a_1, a_2), (a_3, a_4)]$ and $b = [(b_1, b_2), (b_3, b_4)]$ can be performed as follows:

$$ab = [(a_1, a_2)(b_1, b_2) - (a_3, a_4)(b_3, b_4), (a_3, a_4)(b_1, b_2) + (a_1, a_2)(b_3, b_4)].$$

It should be noted that multiplication in $\mathbb{F}_9$ requires multiplication, addition and negation in $\mathbb{F}_3$.

We have implemented the polynomial multiplication of degree $(n-1)$ for $n = 167$, 239, 317, and 353. For each of these values, we have used the proposed algorithms for $\mathbb{F}_3$ and $\mathbb{F}_9$ for a number of recursions at the beginning and as the size of polynomials became smaller we have switched to other algorithms so that the overall arithmetic complexity could be kept at a minimum. Table 2 lists the sequence of algorithms used for each of the values of n. For example, for the multiplication of polynomials of degree 166 (or size 167) over $\mathbb{F}_3$, KA is used for the first six recursions, and SB is then used for the multiplication of polynomials of degree two. For computing the multiplication of polynomials of degree 166 over $\mathbb{F}_9$, we used $A1_9$ twice, KA_9 once, $A2_9$ once, KA once and SB five times.

We have implemented the proposed algorithms at the Register Transfer Level (RTL) using Verilog HDL. For each algorithm, the sequence of operations described in Table 2 has been realized as pure combinational circuits. As an example, a high level block diagram of our circuits for the multiplication algorithm $A1_9$ is given in Appendix. In our implementation using Verilog, the schoolbook multiplication, $\mathbb{F}_3$ addition and $\mathbb{F}_9$ addition have each been coded to be configurable so that they can be instantiated in the recursive tree simply by passing the size parameter. The gate level synthesis has been performed in a Synopsys Design Compiler Version E-2010.12 using the TSMC 65 nm standard cell library at the worst case corner. The synthesis has been targeted to optimize for the area. The total areas and critical path delays achieved in the post-synthesis simulation for a variety of values of n are listed in Table 3. We note that there seems to be no previous ASIC implementation of characteristic three polynomial or field multiplication

Table 3 Implementation results for polynomial multiplication over $\mathbb{F}_3$ and $\mathbb{F}_9$ for special values of n.

n	Multiplication in $\mathbb{F}_3[X]$		Multiplication in $\mathbb{F}_9[X]$	
	Area (μm^2)	Delay (ns)	Area (μm^2)	Delay (ns)
167	253598	6.05	615590	7.19
193	310440	6.37	818432	8.26
239	420115	6.23	1078434	8.68
317	607605	5.49	1631070	8.25
353	841728	7.04	2009400	9.06

algorithms for values of n similar to those reported in Table 3. Readers interested in FPGA implementation results for multiplication of elements over characteristic fields like $\mathbb{F}_{3^{97}}$ are referred to [20].

6 Further Improvements

$M_3(n)$ can be improved about 50% if we design a new algorithm as follows: We use the evaluation points 0, 1, ω, $-\omega$ and ∞. Then we have

$$\begin{aligned}
&\text{Evaluation at } X=0 \Longrightarrow P_0 = A_0B_0 = C_0\\
&\text{Evaluation at } X=1 \Longrightarrow P_1 = (A_0+A_1+A_2)\\
&\quad\times(B_0+B_1+B_2) = C_0+C_1+\cdots+C_4\\
&\text{Evaluation at } X=\omega \Longrightarrow P_2 = (A_0+A_1\omega-A_2)\\
&\quad\times(B_0+B_1\omega-B_2) = C_0+C_1\omega-\cdots+C_4\\
&\text{Evaluation at } X=-\omega \Longrightarrow P_3 = (A_0-A_1\omega-A_2)\\
&\quad\times(B_0-B_1\omega-B_2) = C_0-C_1\omega+\cdots+C_4\\
&\text{Evaluation at } X=\infty \Longrightarrow P_4 = A_2B_2 = C_4.
\end{aligned}$$

Let $P_2 = P_{2,0} + \omega P_{2,1}$ and $P_3 = P_{3,0} + \omega P_{3,1}$. It can be observed that $P_{2,0} = P_{3,0}$ and $P_{2,1} = -P_{3,1}$, which shows that P_3 can be obtained from P_2, thus avoiding the requirement to compute P_3. The following formula and recursions can easily be obtained with the use of the method described previously in this section:

$$\begin{cases}
C_0 = P_0,\\
C_1 = -P_0 - P_1 - P_{2,0} - P_4 + P_{2,1}\omega,\\
C_2 = -P_0 - P_{2,0} + P_4,\\
C_3 = -P_0 - P_1 - P_{2,0} - P_4 - P_{2,1}\omega,\\
C_4 = P_4.
\end{cases} \tag{24}$$

$$\begin{cases}
M_{3,\otimes}(n) \le 3M_{\otimes}(n/3) + M_{9,\otimes}(n/3),\ M_{3,\otimes}(1) = 1,\\
M_{3,\oplus}(n) \le 3M_{3,\oplus}(n/3) + M_{9,\oplus}(n/3) + 14n/3 - 6,\ M_{3,\oplus}(1) = 0,\\
M_3(n) \le 3M_3(n/3) + M_9(n/3) + 14n/3 - 6,\ M_3(1) = 6,\\
D_3(n) \le D_9(n/3) + 7D_{\oplus}.
\end{cases} \tag{25}$$

$$\begin{cases}
M_{3,\otimes}(n) \le 2n^{\log_3 5} - n,\\
M_{3,\oplus}(n) \le 13n^{\log_3 5} - 4.85n\log_3 n - 13n,\\
M_3(n) \le 15n^{\log_3 5} - 4.85n\log_3 n - 14n,\\
D_3(n) \le (7\log_3 n + 1)D_{\oplus} + D_{\otimes}.
\end{cases} \tag{26}$$

The complexity of each algorithm is summarized in Table 4. It should be noted that the new 3-way algorithm outperforms the improved Karatsuba algorithm when $n > 400$.

Example 1 This example illustrates our design of an area efficient algorithm for $n = 709$. Using the recursion in Eq. 25 and the values for $M_3(239)$, $M_9(239)$ and $M_3(355)$ from Table 2 yields the following results:

$$\begin{aligned}
M_3(709) &< M_3(717) \le 3M_3(239)\\
&\quad + M_9(239) + 14\cdot 239 - 6 = 191870.
\end{aligned}$$

Table 4 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)	$6n^{\log_2 3} - 8n + 2$	$n^{\log_2 3}$	$3\log_2 n D_{\oplus} + D_{\otimes}$
Improved 2-way (Section 3.2)	$5.5n^{\log_2 3} - 7n + 1.5$	$n^{\log_2 3}$	$3\log_2 n D_{\oplus} + D_{\otimes}$
3-way (Section 3.3)	$5.8n^{\log_3 6} - 8n + 2.2$	$n^{\log_3 6}$	$4\log_3 n D_{\oplus} + D_{\otimes}$
Improved 3-way (Section 3.4)	$5.53n^{\log_3 6} - 7.33n + 1.8$	$n^{\log_3 6}$	$4\log_3 n D_{\oplus} + D_{\otimes}$
New 3-way (Section 4)	$26n^{\log_3 5} - 33.33n^{\log_3 4} + 6n + 1.33$	$4n^{\log_3 5} - 3n^{\log_3 4}$	$(7\log_3 n + 1)D_{\oplus} + D_{\otimes}$
New 3-way (Section 6)	$13n^{\log_3 5} - 4.85n\log_3 n - 13n$	$2n^{\log_3 5} - n$	$(7\log_3 n + 1)D_{\oplus} + D_{\otimes}$

On the other hand, improved Karatsuba gives

$$M_3(709) < M_3(710) \leq 3M_3(355) + 7 \cdot 355 - 3.$$

Even if we use $M_3(355) = M_3(353) \leq 67761$ from Table 2, we get

$$M_3(709) \leq 205765.$$

The improved Karatsuba thus yields $M_3(709) \leq 205765$ while the new algorithm results in $M_3(709) \leq 191870$, i.e., an approximately 7% reduction in the complexity

Remark 3 The classical approach for performing multiplication in $\mathbb{F}_{3^{2n}}$ is to use the Karatsuba algorithm in the first recursion so that the complexity becomes approximately $M_3(2n) \leq 3M_3(n)$. The other possible method relies on the extension field representation of the elements based on the use of $\mathbb{F}_{3^{2n}} \cong \mathbb{F}_{9^n}$. The elements of $\mathbb{F}_{3^{2n}}$ can then be represented by the polynomials over $\mathbb{F}_9$ of degree less than n. This method requires approximately $M_3(2n) \leq M_9(n)$. Recall that $M_9(n) \approx 2.5M_3(n)$ (see Table 2). The use of the 3-way algorithm for multiplication of polynomials over $\mathbb{F}_9$ is therefore superior to the classical approach by about 15%.

7 Conclusion

In this paper, we have proposed improved algorithms for multiplication in $\mathbb{F}_{3^n}$. As a first step, we introduced improvements to the classical Karatsuba algorithm, which can also be employed for characteristic three fields, and we also indicated the computational cost of the improved Karatsuba 2-way and 3-way algorithms. Next, we explained our derivation of a new 3-way polynomial multiplication algorithm with five 1/3 sized multiplications using interpolation in $\mathbb{F}_9$ and determined the arithmetic and delay complexity associated with the recursive use of this algorithm. We then described ASIC implementation of multiplication of polynomials that are of practical interest. The final contribution of this work is another efficient algorithm for multiplication in $\mathbb{F}_{3^n}$ that uses polynomial multiplication over $\mathbb{F}_9$ and produces superior results. This algorithm leads to about 15% reduction in terms of the number of basic $\mathbb{F}_3$ operations needed for fields considered in Section 5.

Appendix

A high level block diagram of our circuits for the multiplication algorithm of $A1_9$ is presented in Fig. 2. A and B are two degree $(n-1)$ polynomials over $\mathbb{F}_3$. They are split into three parts as $A(X) = A_0 + A_1X^{n/3} + A_2X^{2n/3}$ 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 $n/3$. For the details of the algorithm, we refer the reader to Section 4.

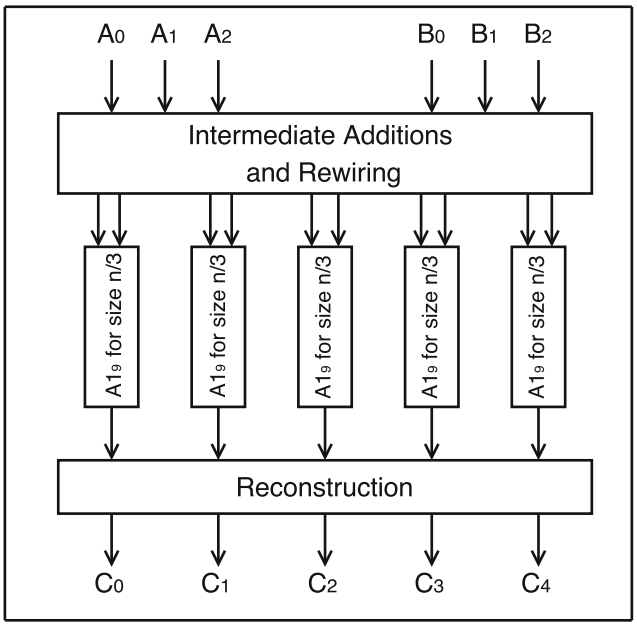

Figure 2 A high level block diagram of $A1_9$.

References

1. Ahmadi, O., Hankerson, D., & Menezes, A. (2007). Formulas for cube roots in F_{3^m}. *Discrete Applied Mathematics*, *155*(3).
2. Ahmadi, O., Hankerson, D., & Menezes, A. (2007). Software implementation of arithmetic in F_{3^m}. In *WAIFI* (pp. 85–102).
3. Barbulescu, R., Detrey, J., Estibals, N., & Zimmermann, P. (2012). Finding optimal formulae for bilinear maps. In *WAIFI* (pp. 168–186).
4. Bernstein, D.J. (2009). Batch binary Edwards. In *Advances in cryptology - CRYPTO 2009, volume 5677 of LNCS* (pp. 317–336).
5. Cenk, M., Koç, Ç.K., & Özbudak, F. (2009). Polynomial multiplication over finite fields using field extensions and interpolation. In *IEEE symposium on computer arithmetic* (pp. 84–91).
6. Cenk, M., Hasan, M.A., & Negre, C. (2014). Efficient subquadratic space complexity binary polynomial multipliers based on block recombination. *IEEE Transactions on Computers*, *63*(9), 2273–2287.
7. Cenk, M., Negre, C., & Hasan, M.A. (2011). Improved three-way split formulas for binary polynomial multiplication. In *Selected areas in cryptography* (pp. 384–398).
8. Cenk, M., Negre, C., & Anwar Hasan, M. (2013). Improved three-way split formulas for binary polynomial and toeplitz matrix vector products. *IEEE Transactions on Computers*, *62*(7), 1345–1361.
9. Cenk, M., & Özbudak, F. (2008). Efficient multiplication in $\mathbb{F}_{3^{\ell m}}$, $m \geq 1$ and $5 \leq \ell \leq 18$. In *AFRICACRYPT* (pp. 406–414).
10. Fan, H., & Hasan, M.A. (2007). A new approach to subquadratic space complexity parallel multipliers for extended binary fields. *IEEE Transactions on Computers*, *56*(2), 224–233.
11. Farashahi, R.R., Wu, H., & Zhao, C. (2013). Efficient arithmetic on elliptic curves over fields of characteristic three. In *Selected areas in cryptography* (pp. 135–148).
12. Von Zur Gathen, J., & Gerhard, J. (2013). *Modern computer algebra*, 3rd edn. Cambridge: Cambridge University Press.

13. Gorla, E., Puttmann, C., & Shokrollahi, J. (2007). Explicit formulas for efficient multiplication in $F_{3^{6m}}$. In *Selected areas in cryptography* (pp. 173–183).
14. Hisil, H., Carter, G., & Dawson, E. (2007). New formulae for efficient elliptic curve arithmetic. In *Progress in cryptology–INDOCRYPT 2007* (pp. 138–151).
15. Karatsuba, A.A., & Ofman, Y. (1963). Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, *7*(2), 595–596.
16. Kim, K.H., Choe, J.S., & Kim, S.I. (2007). New fast algorithms for arithmetic on elliptic curves over finite fields of characteristic three. Cryptology ePrint Archive, Technical Report 2007/179.
17. Koblitz, N. (1998). An elliptic curve implementation of the finite field digital signature algorithm. In *Advances in cryptology–CRYPTO'98* (pp. 327–337). Springer.
18. Montgomery, P.L. (2005). Five, six, and seven-term karatsuba-like formulae. *IEEE Transactions on Computers*, *54*(3), 362–369.
19. Negre, C. (2005). Scalar multiplication on elliptic curves defined over fields of small odd characteristic. In *Progress in cryptology–INDOCRYPT 2005* (pp. 389–402). Springer.
20. Page, D., & Smart, N.P. (2003). Hardware implementation of finite fields of characteristic three. In *Cryptographic hardware and embedded systems-CHES 2002* (pp. 529–539). Springer.
21. Smart, N.P., & Westwood, E.J. (2003). Point multiplication on ordinary elliptic curves over fields of characteristic three. *Applicable Algebra in Engineering, Communication and Computing*, *13*(6), 485–497.
22. Winograd, S. (1980). *Arithmetic complexity of computations*. Society For Industrial & Applied Mathematics, U.S.
23. Zhou, G., & Michalik, H. (2010). Comments on a new architecture for a parallel finite field multiplier with low complexity based on composite field. *IEEE Transactions on Computers*, *59*(7), 1007–1008.

Murat Cenk received the BS degree in mathematics from Middle East Technical University (METU), Ankara, Turkey, in 2000, the MS degree in mathematics and computer science from Cankaya University, Ankara, Turkey, in 2003, and the PhD degree in cryptography from the Institute of Applied Mathematics at METU in 2009. He was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada between September 2010 and January 2014. He then joined to the Institute of Applied Mathematics at METU as a faculty member. His research interests include design and analysis of efficient algorithms for cryptographic applications.

Farhad Haghighi Zadeh received the B.Eng. degree in electrical engineering from Sharif University of Technology, Tehran, Iran in 2010, and the M.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, Canada in 2012. Since then, he has worked at Insight Design Labs (now part of XR Trading), Elliptic Technologies (now part of Synopsys) and Qualcomm. He is currently a senior security R&D engineer at Synopsys, Toronto, Canada.

M. Anwar Hasan received the B.Sc. degree in electrical and electronic engineering, the M.Sc. degree in computer engineering, both from the Bangladesh University of Engineering and Technology, in 1986 and 1988, respectively, and the Ph.D. degree in electrical engineering from the University of Victoria in 1992.

Dr. Hasan joined the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada in 1993 and has been a full professor since 2002. At the university of Waterloo, he is also a member of the Centre for Applied Cryptographic Research, the Center for Wireless Communications, and the VLSI Research group.

Dr. Hasan is a recipient of the Raihan Memorial Gold Medal. At the University of Victoria, he was awarded the President's Research Scholarship four times. At the University of Waterloo, he received a Faculty of Engineering Distinguished Performance Award in 2000, and Outstanding Performance Awards in 2004 and 2010. He served on the program and executive committees of several conferences.

He is a licensed professional engineer of Ontario.