# Improved Three-Way Split Formulas for Binary Polynomial Multiplication

Murat Cenk[1], Christophe Negre[1,2,3], and M. Anwar Hasan[1]

[1] Department of Electrical and Computer Engineering, University of Waterloo, Canada
[2] LIRMM, Université Montpellier 2, France
[3] Team DALI, Université de Perpignan, France

**Abstract.** In this paper we deal with 3-way split formulas for binary field multiplication with five recursive multiplications of smaller sizes. We first recall the formula proposed by Bernstein at CRYPTO 2009 and derive the complexity of a parallel multiplier based on this formula. We then propose a new set of 3-way split formulas with five recursive multiplications based on field extension. We evaluate their complexities and provide a comparison.

## 1 Introduction

Several cryptographic applications like those relying on elliptic curve cryptography [7,9] or Galois Counter Mode [8] require efficient finite field arithmetic. For example, ciphering a message using the ElGamal [3] scheme over an elliptic curve requires several hundreds of multiplications and additions in the finite field.

In this paper we will consider only binary fields. A binary field $\mathbb{F}_{2^n}$ can be viewed as the set of binary polynomials of degree $< n$. An addition of two elements in $\mathbb{F}_{2^n}$ consists of a bitwise XOR of the $n$ coefficients and it can be easily implemented either in software or hardware. The multiplication is more complicated: it consists of a polynomial multiplication and a reduction modulo an irreducible polynomial. The reduction is generally quite simple since the irreducible polynomial can be chosen as a trinomial or pentanomial. The most challenging operation is thus the polynomial multiplication.

The degree $n$ of the field $\mathbb{F}_{2^n}$ used in today's elliptic curve cryptography (ECC) is in the range of $[160, 600]$. For this size of polynomials, recursive methods like Karatsuba [6] or Toom-Cook [11,12] are considered to be most appropriate. Several parallel multipliers have been proposed based on such approaches [10,5]. They all have subquadratic arithmetic complexity, i.e., the number of bit operations is $O(n^{1+\varepsilon})$, where $0 < \varepsilon < 1$, and they are *logarithmic* in time. When such a subquadratic complexity multiplier is implemented in hardware in bit parallel fashion, well known approaches include 2-way split formulas with three recursive multiplications and 3-way split formulas with six recursive multiplications [10,5]. Recently, Bernstein in [1] has proposed a 3-way split formula with only *five* recursive multiplications.

A. Miri and S. Vaudenay (Eds.): SAC 2011, LNCS 7118, pp. 384–398, 2012.

In this paper we also deal with the 3-way splits and propose new formulas for binary polynomial multiplication with five recursive multiplications. We use the extension field $\mathbb{F}_4$ to obtain a sufficient number of elements to be able to apply the multi-evaluation (i.e., evaluation at multiple elements) and interpolation method. This leads to Toom-Cook like formulas. We study the recursive complexity of the proposed formulas and evaluate the delay of the corresponding parallel multiplier.

The remainder of this paper is organized as follows: in Section 2 we review the general method based on multi-evaluation and interpolation to obtain 3-way split formulas. We then review Bernstein's formula and evaluate a non-recursive form of its complexity. In Section 3 we present a new set of 3-way formulas based on field extension. We evaluate the complexity and the delay of a parallel multiplier based on these formulas. Complexity comparison and some concluding remarks are given in Section 4.

## 2 Review of 3-Way Splitting Methods for Polynomial Multiplication

In this section we review the general approach to the design of 3-way split formulas for binary polynomial multiplication. Then we review the 3-way split methods with five multiplications of [1] and study its complexity. Pertinent lemmas along with their proofs are given in Appendix A.

### 2.1 General Approach to Design 3-Way Split Multiplier

A classical method to derive Toom-Cook like formulas consists of applying the multi-evaluation and interpolation approach. Let us consider two degree $n-1$ polynomials $A(X) = \sum_{i=0}^{n-1} a_i X^i$ and $B(X) = \sum_{i=0}^{n-1} b_i X^i$ in $\mathcal{R}[X]$, where $\mathcal{R}$ is an arbitrary ring and $n$ a power of 3. We split $A$ and $B$ in three parts: $A = \sum_{i=0}^{2} A_i X^{in/3}$ and $B = \sum_{i=0}^{2} B_i X^{in/3}$, where $A_i$ and $B_i$ are degree $n/3-1$ polynomials. We replace $X^{n/3}$ by the indeterminate $Y$ in these expressions of $A$ and $B$. We then fix four elements $\alpha_1, \ldots, \alpha_4 \in \mathcal{R}$ plus the infinity element $\alpha_5 = \infty$. Finally we multi-evaluate $A(Y)$ and $B(Y)$ at these five elements and we multiply term by term $A(\alpha_i)$ and $B(\alpha_i)$ for $i = 1, \ldots, 5$, which provides $C(\alpha_i)$ of $C(Y) = A(Y) \times B(Y)$ at those five elements.

These five multiplications can be computed by recursively applying the same process for degree $n/3-1$ polynomial in $X$. We then interpolate $C(Y)$ to obtain its polynomial expression in $Y$. Specifically, if we define the Lagrange polynomial as $L_i(Y) = \prod_{j=1, j\neq i}^{4} \left(\frac{Y-\alpha_j}{\alpha_i-\alpha_j}\right)$ for $i = 1, \ldots, 4$ and $L_\infty = \prod_{i=1}^{4}(Y-\alpha_i)$ then we have

$$C(Y) = \sum_{i=1}^{4} L_i(Y)C(\alpha_i) + C(\infty)L_\infty(Y).$$

We obtain the regular expression of $C$ as a polynomial in $X$ by replacing $Y$ by $X^{n/3}$.

### 2.2 Bernstein's 3-Way Split Formula

In this subsection, first we review the 3-way split formula with five recursive multiplications presented by Bernstein in [1]. We then derive its complexity results. We consider two degree $n-1$ polynomials $A$ and $B$ in $\mathbb{F}_2[X]$ where $n$ is a power of 3. We split these two polynomials in three parts and then replace $X^{n/3}$ by $Y$ and consider them as polynomial in $\mathcal{R}[Y]$ where $\mathcal{R} = \mathbb{F}_2[X]$

$$A = A_0 + A_1 Y + A_2 Y^2 \text{ and } B = B_0 + B_1 Y + B_2 Y^2$$

with $\deg_X A_i, \deg_X B_i < n/3$. Bernstein uses a multi-evaluation and interpolation approach by evaluating the polynomials at these five elements $1, 0, X, X+1$ and $\infty$ of $\mathcal{R} \cup \{\infty\}$. We denote $C$ as the product of $A$ and $B$. We then define the pairwise products of the evaluations of $A(Y)$ and $B(Y)$ at $0, 1, X, X+1$ and $\infty$ as follows

$$\begin{aligned}
&P_0 = A_0 B_0 \quad \text{(eval. at 0)},\\
&P_1 = (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) \quad \text{(eval. at 1)},\\
&P_2 = (A_0 + A_1 X + A_2 X^2)(B_0 + B_1 X + B_2 X^2) \quad \text{(eval. at } X\text{)},\\
&P_3 = \big((A_0 + A_1 + A_2) + (A_1 X + A_2 X^2)\big)\\
&\qquad\qquad \times \big((B_0 + B_1 + B_2) + (B_1 X + B_2 X^2)\big) \quad \text{(eval. at } X+1\text{)},\\
&P_4 = A_2 B_2 \quad \text{(eval. at } \infty\text{)}.
\end{aligned}$$

Bernstein has proposed the following expressions for the reconstruction of $C$:

$$\begin{aligned}
&U = P_0 + (P_0 + P_1)X \text{ and } V = P_2 + (P_2 + P_3)(X^{n/3} + X), \text{ then}\\
&C = U + P_4(X^{4n/3} + X^{n/3}) + \frac{(U + V + P_4(X^4 + X))(X^{2n/3} + X^{n/3})}{X^2 + X}. \qquad (1)
\end{aligned}$$

### 2.3 Asymptotic Complexity of the Bernstein Method

We evaluate the complexity of the formula of Bernstein when they are applied recursively. This complexity will be expressed in terms of the number of bit addition denoted as $\mathcal{S}_{\oplus}(n)$ and the number of bit multiplication denoted $\mathcal{S}_{\otimes}(n)$. The complexities of the computation of the five products $P_0, P_1, P_2, P_3$ and $P_4$ are given in Table 7 in Appendix B. Note that the degrees of $R_3, R_3', R_4$ and $R_4'$ are all equal to $n/3+1$, while the degrees of $A_0, B_0, A_2, B_2, R_1$ and $R_1'$ are equal to $n/3-1$. Consequently, the products $P_0$, $P_1$ and $P_4$ have their degree equal to $(2n/3-2)$ and the degrees of $P_2$ and $P_3$ are equal to $(2n/3+2)$.

The formulas in Table 7 can be applied only once since the five products involve polynomials of degree $n/3-1$ and $n/3+1$. In order to have a fully recursive method, we express the product of degree $n/3+1$ polynomials in terms of one product of degree $n/3-1$ polynomial plus some additional non-recursive computations.

For this purpose, we consider $P = \sum_{i=0}^{n/3+1} p_i X^i$ and $Q = \sum_{i=0}^{n/3+1} q_i X^i$. We first rewrite $P$ as $P = P' + (p_{n/3} X^{n/3} + p_{n/3+1} X^{n/3+1})$ and $Q = Q' + (q_{n/3} X^{n/3} + q_{n/3+1} X^{n/3+1})$ and then if we expand the product $PQ$ we obtain

$$PQ = \underbrace{P'Q'}_{M_1} + \underbrace{(p_{n/3}X^{n/3} + p_{n/3+1}X^{n/3+1})Q'}_{M_2} + \underbrace{(q_{n/3}X^{n/3} + q_{n/3+1}X^{n/3+1})P'}_{M_3} + \underbrace{(p_{n/3}X^{n/3} + p_{n/3+1}X^{n/3+1})(q_{n/3}X^{n/3} + q_{n/3+1}X^{n/3+1})}_{M_4}. \quad (2)$$

The product $M_1$ can be performed recursively since $P'$ and $Q'$ are of degree $n/3-1$ each. The other products $M_2, M_3$ and $M_4$ can be computed separately and then added to $M_1$. The computation of $M_2$ and $M_3$ is not difficult, each consisting of $2n/3$ bit multiplications and $n/3-1$ bit additions. We compute the product $M_4$ as follows

$$M_4 = p_{n/3}q_{n/3}X^{2n/3} + (p_{n/3+1}q_{n/3} + p_{n/3}q_{n/3+1})X^{2n/3+1} + p_{n/3+1}q_{n/3+1}X^{2n/3+2}$$

and this requires one bit additions and four bit multiplications. Finally, the complexities $\mathcal{S}_\oplus(n/3)$ and $\mathcal{S}_\otimes(n/3)$ consist of the complexity of each product $M_i$ plus $2n/3+1$ bit additions for the sum of these five products. This results in the following complexity:

$$\begin{cases} \mathcal{S}_\oplus(n/3+2) = \mathcal{S}_\oplus(n/3) + 4n/3, \\ \mathcal{S}_\otimes(n/3+2) = \mathcal{S}_\otimes(n/3) + 4n/3 + 4. \end{cases} \quad (3)$$

*Explicit computations of the reconstruction.* We now review the sequence of computation proposed by Bernstein in [1] for the reconstruction. This sequence first computes the two polynomials $U$ and $V$ defined in (1) and then computes $C$. The details are given in Table 8 in Appendix B.

With regard to the division of $W = (U + V + P_4(X^4 + X))(X^{2n/3} + X^{n/3})$ by $X^2 + X$ in the reconstruction (1) (i.e., the computation of $W'$ in Table 8), we remark that $W$ is of degree $n$, so we can write $W = w_nX^n + \ldots + w_1X + w_0$. Since $X^2 + X = X(X + 1)$, the division can be performed in two steps: first we divide $W$ by $X$ which consists of a shift of the coefficients of $W$ and then we divide $W/X = w_nX^{n-1} + \ldots + w_1$ by $X + 1$. The result $W' = W/(X^2 + X)$ has its coefficients defined as follows:

$$w'_{n-j} = w_n + w_{n-1} + \cdots + w_{n-j+2}.$$

These computations require $n-2$ bit additions: we perform sequentially the additions $w'_i = w'_{i+1} + w_{i+2}$ starting from $w'_{n-2} = w_n$. The corresponding delay is then equal to $(n-2)D_\oplus$ where $D_\oplus$ is the delay of a bit addition.

*Overall Arithmetic Complexity* Now we evaluate the overall complexity of Bernstein's method (Table 7 and 8 in Appendix B). By adding the number of bit additions listed in the two tables in Appendix B we obtain $\mathcal{S}_\oplus(n) = 3\mathcal{S}_\oplus(n/3) + 2\mathcal{S}_\oplus(n/3+2) + 35n/3 - 12$ and for the bit multiplication we have $\mathcal{S}_\otimes(n) = 3\mathcal{S}_\otimes(n/3) + 2\mathcal{S}_\otimes(n/3+2)$. In order to obtain a recursive expression of the complexity, we replace $\mathcal{S}_\oplus(n/3+2)$ and $\mathcal{S}_\otimes(n/3+2)$ by their corresponding expression in terms of $\mathcal{S}_\oplus(n/3)$ and $\mathcal{S}_\otimes(n/3)$ given in (3). We then obtain the following:

$$\mathcal{C} = \begin{cases} \mathcal{S}_\oplus(n) = 5\mathcal{S}_\oplus(n/3) - \frac{43n}{3} - 12, \\ \mathcal{S}_\otimes(n) = 5\mathcal{S}_\otimes(n/3) + \frac{8n}{3} + 8. \end{cases}$$

Then we apply Lemma 1 from Appendix A and we obtain the following:

$$\mathcal{C} = \begin{cases} \mathcal{S}_{\oplus}(n) = \frac{37}{2} n^{\log_3(5)} - \frac{43n}{2} + 3, \\ \mathcal{S}_{\otimes}(n) = 7n^{\log_3(5)} - 4n - 2. \end{cases} \tag{4}$$

*Delay of parallel computation based on Bernstein's method* Here we evaluate the delay of a parallel multiplier based on Bernstein's formula. We will denote $\mathcal{D}(n)$ the delay required for a multiplication of two degree $n-1$ polynomial where $n$ is a power of 3. The delay will be expressed in terms of the delay of bit addition denoted as $D_{\oplus}$ and the delay of bit multiplication denoted as $D_{\otimes}$. For this, we have drawn a data-flow graph of the multi-evaluation and the reconstruction part of the computation. These graphs are shown in Figure 1, from which we remark that the critical path delay is $\mathcal{D}(n/3) + (n+8)D_{\oplus}$. For example, this is the delay of the critical path which starts from $A_0$ or $A_1$ exiting at $R_4$ in the multi-evaluation, then goes through a multiplier of polynomial of degree $n/3+1$ which has a delay of $\mathcal{D}(n/3)+1$ (cf. (2)), and finally enters the reconstruction in $P_3$ and ends at $C$. Since we have assumed that $n$ is a power of 3, we transform

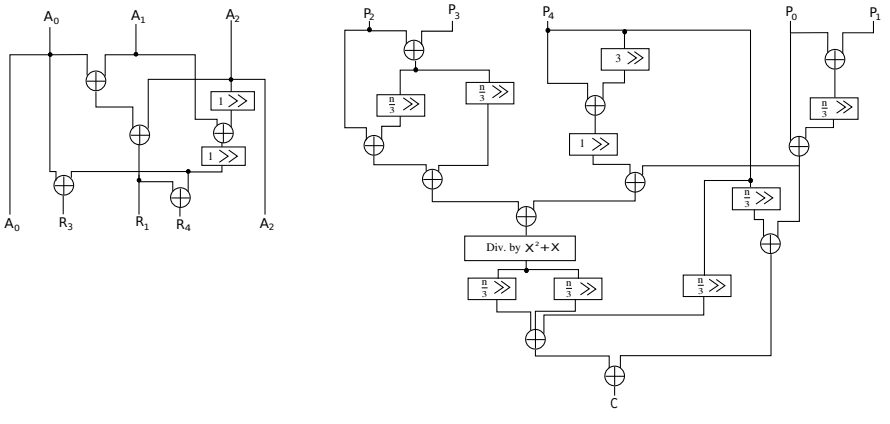


**Fig. 1.** Multi-evaluation (left) and reconstruction (right) data flow

$\mathcal{D}(n) = \mathcal{D}(n/3) + (n+8)D_{\oplus}$ into a non-recursive expression, by applying it recursively and using $\mathcal{D}(1) = D_{\otimes}$.

$$\begin{aligned} \mathcal{D}(n) &= (n+8)D_{\oplus} + (n/3+8)D_{\oplus} + (n/9+8)D_{\oplus} + \ldots + (3+8)D_{\oplus} + D_{\otimes} \\ &= (8\log_3(n) + \tfrac{3n}{2} - \tfrac{3}{2})D_{\oplus} + D_{\otimes}. \end{aligned} \tag{5}$$

## 3 Three-Way Formulas Based on Field Extension

In this section we present an approach based on field extension which provides 3-way split formulas with five recursive multiplications. We consider two binary polynomials $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$ with $n = 3^{\ell}$. As

before, we split $A$ and $B$ in three parts $A = A_0 + A_1X^{n/3} + A_2X^{2n/3}$ and $B = B_0 + B_1X^{n/3} + B_2X^{2n/3}$ where $A_i$ and $B_i$ have degree $< n/3$. We would like to use the approach based on multi-evaluation at five elements reviewed in Subsection 2.1. The problem we faced is that there are not enough elements in $\mathbb{F}_2$: we can only evaluate at the two elements of $\mathbb{F}_2$ and at infinity. Bernstein used the two elements $X$ and $X + 1$ in order to overcome this problem. We use here a different approach: in order to evaluate at two more elements we will consider the method proposed in [2,12] which uses a field extension. Specifically, we consider an extension $\mathbb{F}_4 = \mathbb{F}_2[\alpha]/(\alpha^2 + \alpha + 1)$ of degree 2 of $\mathbb{F}_2$. Afterwards, we evaluate the polynomials at 0, 1, $\alpha$, $\alpha + 1$ and $\infty$. The resulting evaluations and recursive multiplication are given below:

$$\begin{array}{ll} P_0 = A_0B_0 & \text{in } \mathbb{F}_2[X], \\ P_1 = (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) & \text{in } \mathbb{F}_2[X], \\ P_2 = (A_0 + A_2 + \alpha(A_1 + A_2))(B_0 + B_2 + \alpha(B_1 + B_2)), & \text{in } \mathbb{F}_4[X], \\ P_3 = (A_0 + A_1 + \alpha(A_1 + A_2))(B_0 + B_1 + \alpha(B_1 + B_2)), & \text{in } \mathbb{F}_4[X], \\ P_4 = A_2B_2 & \text{in } \mathbb{F}_2[X]. \end{array} \tag{6}$$

The reconstruction of $C = A \times B$ uses the classical Lagrange interpolation. An arranged form of this interpolation is given below

$$\begin{array}{l} C = (P_0 + X^{n/3}P_4)(1 + X^n) + (P_1 + (1 + \alpha)(P_2 + P_3))(X^{n/3} + X^{2n/3} + X^n) \\ \quad +\alpha(P_2 + P_3)X^n + P_2X^{2n/3} + P_3X^{n/3} \end{array} \tag{7}$$

Note that if we evaluate a binary polynomial at $0, 1$ or $\infty$ we obtain a polynomial in $\mathbb{F}_2[X]$ and on the other hand if we evaluate the same polynomial at $\alpha$ or $\alpha+1$ we obtain a polynomial in $\mathbb{F}_4[X]$. These multiplications are performed recursively by splitting and evaluating at the same set of points recursively. We will give a sequence of computations for (6) and (7) dealing with the two following cases: the first case is when the formulas are applied to $A$ and $B$ in $\mathbb{F}_4[X]$ and the second case is when $A$ and $B$ are in $\mathbb{F}_2[X]$.

### 3.1 Explicit 3-Way Splitting Formulas

In this subsection, we provide a sequence of computations for (6) and (7) when $A$ and $B$ are taken in $\mathbb{F}_4[X]$ or in $\mathbb{F}_2[X]$. We split the computations of (6) and (7) in the three different steps . We first give the formulas for the multi-evaluation, then for the products and finally for the reconstruction.

*Multi-evaluation formulas.* The proposed steps to compute the multi-evaluation of $A$ and $B$ are detailed in Table 1. The formulas are the same for polynomials in $\mathbb{F}_4[X]$ and in $\mathbb{F}_2[X]$. The only difference between these two cases is the cost of each computation. For the evaluation of the cost of each operation in $\mathbb{F}_4[X]$ we have used the following facts:

- A sum of two elements of $\mathbb{F}_4$ is given by $(a_0 + a_1\alpha) + (b_0 + b_1\alpha) = (a_0 + b_0) + (a_1 + b_1)\alpha$ and requires 2 bit additions. Consequently, the sum of two degree $d - 1$ polynomials in $\mathbb{F}_4[X]$ requires $2d$ bit addition.

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

| Computations | | Cost in $\mathbb{F}_4$ | Cost in $\mathbb{F}_2$ |
|---|---|---|---|
| | | $\#\oplus$ | $\#\oplus$ |
| $R_1 = A_0 + A_1$, | $R'_1 = B_0 + B_1$ | $4n/3$ | $2n/3$ |
| $R_2 = A_1 + A_2$, | $R'_2 = B_1 + B_2$ | $4n/3$ | $2n/3$ |
| $R_3 = \alpha R_2$, | $R'_3 = \alpha R'_2$ | $2n/3$ | 0 |
| $R_4 = R_1 + R_3$ $(= A(\alpha + 1))$, | $R'_4 = R'_1 + R'_3$ | $4n/3$ | 0 |
| $R_5 = R_4 + R_2$ $(= A(\alpha))$, | $R'_5 = R'_4 + R'_2$ | $4n/3$ | $2n/3$ |
| $R_6 = R_1 + A_2$ $(= A(1))$, | $R'_6 = R'_1 + B_2$ | $4n/3$ | $2n/3$ |
| Total | | $22n/3$ | $8n/3$ |

**Table 2.** Cost of products for the new three-way split formulas

| Computations | Cost in $\mathbb{F}_4$ | | Cost in $\mathbb{F}_2$ | |
|---|---|---|---|---|
| | $\#\oplus$ | $\#\otimes$ | $\#\oplus$ | $\#\otimes$ |
| $P_0 = A_0 B_0$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$ |
| $P_1 = R_6 R'_6$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$ |
| $P_2 = R_5 R'_5$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ |
| $P_3 = R_4 R'_4$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ |
| $P_4 = A_2 B_2$ | $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ | $\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$ |
| Total | $5S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$ | $5S_{\mathbb{F}_4,\otimes}(\frac{n}{3})$ | $3S_{\mathbb{F}_2,\oplus}(\frac{n}{3}) + 2S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$ | $3S_{\mathbb{F}_2,\otimes}(\frac{n}{3}) + 2S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$ |

- The multiplication of an element $a = a_0 + a_1\alpha$ in $\mathbb{F}_4$ by $\alpha$: it is given by $a\alpha = a_1 + (a_0 + a_1)\alpha$ and thus requires one bit additions. This implies that the multiplication of a degree $d-1$ polynomial of $\mathbb{F}_4[X]$ by $\alpha$ requires $d$ bit additions.

When $A$ and $B$ are taken in $\mathbb{F}_2[X]$, we use the following facts to save some computations

- Since the additions performed for $R_1, R_2, R'_1$ and $R'_2$ involve polynomials in $\mathbb{F}_2[X]$ with degree $n/3-1$, they all require $n/3$ bit additions.
- For $R_3$ (resp. $R'_3$), the multiplication of $R_2$ (resp. $R'_2$) by $\alpha$ is free since the coefficients of the polynomial $R_2$ (resp. $R'_2$) are in $\mathbb{F}_2$.
- The addition in $R_4$ (resp. $R'_4$) involves a polynomial with coefficients in $\mathbb{F}_2$ and a polynomial with coefficients in $\alpha\mathbb{F}_2$; it is thus free of any bit operation.
- The operation in each of $R_5, R'_5, R_6$ and $R'_6$ is an addition of polynomial in $\mathbb{F}_2[X]$ with a polynomial in $\mathbb{F}_4[X]$ and thus no bit additions are required for the coefficient corresponding to $\alpha$.

Using these facts and also using that $A_i$ and $B_i$ are degree $n/3-1$ polynomials and $P_i$ is a degree $2n/3-2$ polynomial, we evaluate each step of Table 1 and then deduce the complexity of the multi-evaluation by adding the cost of each step.

*Recursive products.* In Table 2 we give the cost of the five recursive products. In the case of a multiplication in $\mathbb{F}_4[X]$, all the polynomials are in $\mathbb{F}_4[X]$ and

**Table 3.** Three-way split formulas - Reconstruction

| Reconstruction in $\mathbb{F}_4$ | |
|---|---|
| Computations | $\#\oplus$ |
| $U_1 = P_2 + P_3$ | $4n/3 - 2$ |
| $U_2 = \alpha U_1 \quad (= \alpha(P_2 + P_3))$ | $2n/3 - 1$ |
| $U_3 = (1+\alpha)U_1 \quad (= (1+\alpha)(P_2 + P_3))$ | 0 |
| $U_4 = P_1 + U_3 \quad (= P_1 + (1+\alpha)(P_2 + P_3))$ | $4n/3 - 2$ |
| $U_5 = U_4(X^{n/3} + X^{2n/3} + X^{3n/3})$ | $4n/3 - 4$ |
| $U_6 = P_0 + X^{n/3}P_4 \quad (= P_0 + X^{n/3}P_4)$ | $2n/3 - 2$ |
| $U_7 = U_6(1 + X^n) \quad (= P_0 + X^{n/3}P_4)(1 + X^n))$ | 0 |
| $C = U_7 + U_5 + X^n U_2$ $+P_2 X^{2n/3} + P_3 X^{n/3}$ | $20n/3 - 10$ |
| Total | $36n/3 - 21$ |

| Reconstruction in $\mathbb{F}_2$ | |
|---|---|
| Computations | $\#\oplus$ |
| $U_1 = P_2 + P_3$ | $4n/3 - 2$ |
| $U_2 = [\alpha U_1]_{cte}$ | 0 |
| $U_3 = [(1+\alpha)U_1]_{cte}$ | $2n/3 - 1$ |
| $U_4 = [P_1 + U_3]_{cte}$ | $2n/3 - 1$ |
| $U_5 = [U_4(X^{n/3} + X^{2n/3} + X^{3n/3})]_{cte}$ | $2n/3 - 2$ |
| $U_6 = [P_0 + X^{n/3}P_4]_{cte}$ | $n/3 - 1$ |
| $U_7 = [U_6(1 + X^n)]_{cte}$ | 0 |
| $C = [U_7 + U_5 + X^n U_2]_{cte}$ $+P_2 X^{2n/3} + P_3 X^{n/3}]_{cte}$ | $10n/3 - 5$ |
| Total | $21n/3 - 12$ |

thus the cost of the recursive products are $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ and $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$. For the multiplication in $\mathbb{F}_2[X]$, there are three products which involve polynomials in $\mathbb{F}_2[X]$ incurring a cost of $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ and $\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$; the two other products are in $\mathbb{F}_4[X]$ and thus the corresponding cost is $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ and $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$.

*Reconstruction.* In Table 3 we give the sequence of computations for the reconstruction of the product $C$. For the computation in $\mathbb{F}_4[X]$, we evaluate the cost of each computation by using the same facts as in the multi-evaluation computations. For the computation in $\mathbb{F}_2$, since the resulting polynomial $C$ is in $\mathbb{F}_2[X]$ we can save some computations. We use the following facts:

- $P_3$ and $P_4$ are degree $2n/3 - 2$ polynomials in $\mathbb{F}_4[X]$.
- $P_1, P_2$ and $P_5$ are degree $2n/3 - 2$ polynomials in $\mathbb{F}_2[X]$; so we don't need to add their bits corresponding to $\alpha$.
- The polynomial $C$ is in $\mathbb{F}_2[X]$; consequently, we do not need to compute the coefficients corresponding to $\alpha$. Indeed, if $a = a_0 + a_1\alpha$ and $b = b_0 + b_1\alpha$ we denote $[a + b]_{cte} = a_0 + b_0$ which requires only one bit addition. We use the same notation for the polynomials.

### 3.2 Complexity Evaluation

We now evaluate the complexity of the formulas given in Tables 1, 2 and 3. We first evaluate the complexity for a multiplication in $\mathbb{F}_4[X]$.

*Complexity of the formulas in* $\mathbb{F}_4[X]$ We obtain the following complexities in terms of the number of bit additions and multiplications

$$\begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) = 5\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3) + 58n/3 - 21, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) = 5\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3). \end{cases} \tag{8}$$

Now, in order to derive a non-recursive expression of the complexity, we need to know the cost of a multiplication in $\mathbb{F}_4$. There are two ways to perform such multiplication:

- The first method computes the product of $a = a_0 + a_1\alpha$ and $b = b_0 + b_1\alpha$ as follows:

$$(a_0 + a_1\alpha) \times (b_0 + b_1\alpha) = a_0b_0 + (a_0b_1 + a_1b_0)\alpha + a_1b_1(1 + \alpha).$$

  This requires 3 bit additions and 4 bit multiplications.
- The second method computes the product of $a = a_0 + a_1\alpha$ and $b = b_0 + b_1\alpha$ as follows:

$$(a_0 + a_1\alpha) \times (b_0 + b_1\alpha) = (a_0 + a_1)(b_0 + b_1)\alpha + (a_0b_0 + a_1b_1)(1 + \alpha).$$

  This requires 4 bit additions and 3 bit multiplications.

The choice among these methods depends on the relative cost of a bit addition compared to that of a bit multiplication. If the bit multiplication is cheaper, then the first method is advantageous, otherwise it is the second method.

Using Lemma 1 for (8) with the initial condition $\mathcal{S}_{\mathbb{F}_4,\oplus}(1) = 3$ and $\mathcal{S}_{\mathbb{F}_4,\otimes}(1) = 4$, the first method leads to the complexity $\mathcal{C}$ below. Similarly, using Lemma 1 for (8) with the initial condition $\mathcal{S}_{\mathbb{F}_4,\oplus}(1) = 4$ and $\mathcal{S}_{\mathbb{F}_4,\otimes}(1) = 3$, the second method leads to the complexity $\mathcal{C}'$ below.

$$\mathcal{C} = \begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) = \frac{107}{4}n^{\log_3(5)} - 29n + \frac{21}{4}, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) = 4n^{\log_3(5)}. \end{cases} \mathcal{C}' = \begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) = \frac{111}{4}n^{\log_3(5)} - 29n + \frac{21}{4}, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) = 3n^{\log_3(5)}. \end{cases} \tag{9}$$

*Complexity of recursive 3-way splitting multiplication in* $\mathbb{F}_2[X]$ We evaluate now the overall complexity of the proposed three-way split formulas for polynomials in $\mathbb{F}_2[X]$. If we add the complexity results given in Tables 1, 2 and 3, we obtain the number of bit additions and bit multiplications expressed in terms of $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ and $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ as follows

$$\begin{aligned} \mathcal{S}_{\mathbb{F}_2,\oplus}(n) &= 2\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3) + 3\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3) + 29n/3 - 12, \\ \mathcal{S}_{\mathbb{F}_2,\otimes}(n) &= 2\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3) + 3\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3). \end{aligned} \tag{10}$$

We now derive a non-recursive expression from the previous equation. This is done in two steps: we first replace $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ and $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ by their corresponding non-recursive expression, then we solve the resulting recursive expression of $\mathcal{S}_{\mathbb{F}_2,\oplus}(n)$ and $\mathcal{S}_{\mathbb{F}_2,\otimes}(n)$. We can replace $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ and $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ by the non-recursive expressions $\mathcal{C}$ or $\mathcal{C}'$ given in (9). To this effort, since these computations are essentially identical, we only treat in detail the computation of $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$. In (10), we replace $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ by its expression given in (9) and we obtain $\mathcal{S}_{\mathbb{F}_2,\oplus}(n) = 3\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3) + \frac{107}{10}n^{\log_3(5)} - \frac{29}{3}n - 3/2$. Then a direct application of Lemma 2 from Appendix A yields a non-recursive expression as follows: $\mathcal{S}_{\mathbb{F}_2,\oplus}(n) = \frac{107}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) - \frac{55n}{2} + \frac{3}{4}$.

We then apply the same method to other complexities. Below we list the final non-recursive expression for each case.

$$\mathcal{C} = \begin{cases} \mathcal{S}_{\mathbb{F}_2,\oplus}(n) = \frac{107}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) \\ \qquad\qquad -\frac{55n}{2} + \frac{3}{4} \\ \mathcal{S}_{\mathbb{F}_2,\otimes}(n) = 4n^{\log_3(5)} - 3n \end{cases} \qquad (11)$$

$$\mathcal{C}' = \begin{cases} \mathcal{S}_{\mathbb{F}_2,\oplus}(n) = \frac{111}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) \\ \qquad\qquad -\frac{57n}{2} + \frac{3}{4} \\ \mathcal{S}_{\mathbb{F}_2,\otimes}(n) = = 3n^{\log_3(5)} - 2n \end{cases}$$

### 3.3 Delay Evaluation

We evaluate the delay of the 3-way split multiplier by drawing the data flow of the 3-way multiplier in $\mathbb{F}_4[X]$ and in $\mathbb{F}_2[X]$. The sequence of operations for these two cases ($\mathbb{F}_4$ and $\mathbb{F}_2$) are essentially the same: their only difference is on the reconstruction: in the $\mathbb{F}_2[X]$ multiplication the operations are restricted to $\mathbb{F}_2$. The data flow shown in Figure 2 is valid for both cases. We now evaluate the critical path delay for the multiplication in $\mathbb{F}_4[X]$ and then for the multiplication in $\mathbb{F}_4[X]$.

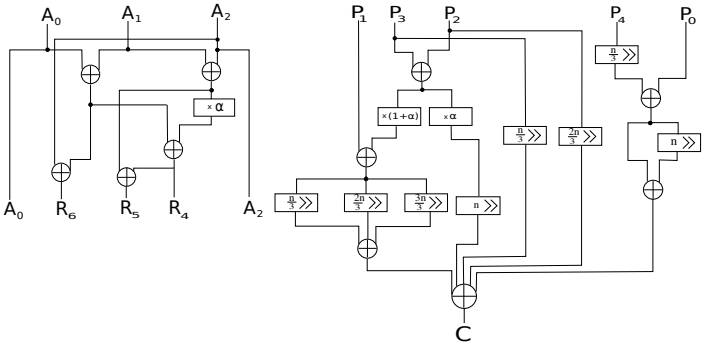


**Fig. 2.** Multi-evaluation (left) and reconstruction (right) data flow

*Delay of the multiplier in* $\mathbb{F}_4[X]$. The critical path is made of the following three parts:

- The critical path in the multi-evaluation data-flow begins in $A_2$, goes through three $\oplus$'s and one multiplication by $\alpha$ and then ends in $R_1$. Since a multiplication by $\alpha$ consists of one bit addition, the delay of this critical path is $4D_\oplus$.
- The path goes through a multiplier for degree $n/3-1$ polynomials with a delay of $\mathcal{D}_{\mathbb{F}_4}(n/3)$.
- Finally, in the reconstruction part, the path enters the reconstruction in $P_2$ and goes through one multiplication by $(1+\alpha)$ and three additions and then in a multi-input $\oplus$. A careful observation of this last multi-input addition shows that the delay in terms of the 2-input $\oplus$ gate is $2D_\oplus$. Consequently, the critical path delay of this part is $6D_\oplus$.

By summing up the above three delay components, we obtain a recursive expression of the delay as $\mathcal{D}_{\mathbb{F}_4}(n) = 10D_X + \mathcal{D}_{\mathbb{F}_4}(n/3)$. After solving this inductive relation, we obtain the following non-recursive expression:

$$\mathcal{D}_{\mathbb{F}_4}(n) = (10\log_3(n) + 2)D_\oplus + D_\otimes. \tag{12}$$

*Delay of the multiplier in* $\mathbb{F}_2[X]$. The critical path is the same as the critical path for the multiplication in $\mathbb{F}_4[X]$. The only difference is that the multiplication by $\alpha$ and $(1+\alpha)$ does not give any delay since it consists of some permutation of the coefficients. Consequently, the recursive expression of the delay is $\mathcal{D}_{\mathbb{F}_2}(n) = 8D_X + \mathcal{D}_{\mathbb{F}_4}(n/3)$ and this yields the corresponding non-recursive expression to be

$$\mathcal{D}_{\mathbb{F}_2}(n) = 10\log_3(n)D_\oplus + D_\otimes. \tag{13}$$

## 4 Complexity Comparison and Conclusion

In this paper, we have first reviewed Bernstein's recently proposed formula for polynomial multiplication using the 3-way split that requires five recursive multiplications. We have carefully evaluated its cost and have provided a non-recursive form of its complexity. We have then presented a new set of 3-way split formulas for binary polynomial multiplication based on field extension. For the proposed formulas, we have computed two non-recursive forms of the complexity: one minimizes the number of bit additions and the other minimizes the number of bit multiplications. We have also evaluated the time delays of parallel multipliers based on the proposed formulas.

Assuming that $n$ is a power of three, the resulting complexities of Bernstein's and the proposed formulas are reported in Table 4. As it can be seen from Table 4, in the asymptotic sense, the ratio of the total number of bit level operations (addition and multiplication combined) of the Bernstein formula and that of either of the proposed formulas is close to $\frac{(18.5+7)}{(26.75+4)} \cong 0.82$. We can also remark that the proposed method are less expensive in term of bit addition,

consequently if the cost of a bit multiplication is twice the cost of a bit addition then the complexity of the proposed method become smaller than the one of Bernstein approach. On the other hand, when those formulas are applied to parallel implementation for polynomial multipliers, Bernstein's formula leads to a time delay linear in $n$, while the proposed ones are logarithmic.

**Table 4.** Complexities of the three approaches considered in this article

| Algorithm | $\mathcal{S}_{\oplus}(n)$ | $\mathcal{S}_{\otimes}(n)$ | Delay |
|---|---|---|---|
| Bernstein [1]) ($\mathcal{C}$ in (4) | $18.5n^{\log_3(5)} - 21.5n + 3$ | $7n^{\log_3(5)} - 4n - 2$ | $(1.5n + 8\log_3(n) - 1.5)D_{\oplus}$ $+D_{\otimes}$ |
| $\mathcal{C}$ from (11) | $26.75n^{\log_3(5)} - 9.67n\log_3(n)$ $-27.5n + 0.75$ | $4n^{\log_3(5)} - 3n$ | $10\log_3(n)D_{\oplus} + D_{\otimes}$ |
| $\mathcal{C}'$ from (11) | $27.75n^{\log_3(5)} - 9.67n\log_3(n)$ $-28.5n + 0.75$ | $3n^{\log_3(5)} - 2n$ | $10\log_3(n)D_{\oplus} + D_{\otimes}$ |

*Improvement for multiplication of polynomials of size $n = 2^i \cdot 3^j$.* Our proposed method can also be used to design efficient multipliers for more generic values of $n$. To illustrate this point, we now consider the situation where we want to perform $A \times B$ where $A$ and $B$ are of degree $n-1$ where $n = 2 \cdot 3^j$. A direct approach to perform this operation consists of first applying the Karatsuba formula which breaks this multiplication into three multiplications of polynomials of degree $3^j - 1$. If these three multiplications are performed using Bernstein's approach, then the cost of $A \times B$ is 3 times the complexity of one instance of Bernstein's approach plus $7n/2 - 3$ bit additions for Karatsuba.

This can be done more efficiently as follows. We perform this multiplication by first splitting the two polynomials in two parts $A = A_0 + X^{n/2}A_1$ and $B = B_0 + X^{n/2}B_1$. We then perform $A_0 \times (B_0 + \alpha B_1)$ and $A_1 \times (B_0 + \alpha B_1)$ using the proposed formulas for multiplication in $\mathbb{F}_4[X]$ of Subsection 3.1. This provides the four products $A_iB_j$ for $i, j \in \{0, 1\}$. The product $C = A \times B$ is then reconstructed as $C = A_0C_0 + X^{n/2}(A_0B_1 + A_1B_0) + B_1A_1X^n$. The cost of this approach is thus two times the cost a degree $n/3 - 1$ multiplications in $\mathbb{F}_4[X]$ plus $2n$ bit additions for the reconstruction. The resulting complexities are reported in Table 5.

**Table 5.** Complexities of a multiplication of polynomials of size $n = 2 \cdot 3^j$

| Method | $\mathcal{S}_{\oplus}(n)$ | $\mathcal{S}_{\otimes}(n)$ |
|---|---|---|
| Karatsuba and Bernstein [1] ($\mathcal{C}$ in (4)) | $20.1n^{\log_3(5)} - 35.75n + 6$ | $7.6n^{\log_3(5)} - 6n - 6$ |
| With $\mathcal{C}$ from Subsection 3.1 | $19.37n^{\log_3(5)} - 29n + 10.5$ | $2.17n^{\log_3(5)} - 3n$ |
| With $\mathcal{C}'$ from Subsection 3.1 | $20.1n^{\log_3(5)} - 29n + 10.5$ | $2.89n^{\log_3(5)} - 2n$ |

*Explicit complexity for polynomial multiplication with practical size.* In Table 6 we give complexity results of polynomial multiplication for $n = 2^i \cdot 3^j$ with cryptographic sizes, specifically, in the range $[160, 500]$. The complexities correspond

to the combination of Karatsuba (cf [1]) and the formula of Bernstein or the proposed formulas. To get the complexity based on the proposed formulas in the special case where $i \geq 1$ and $j \geq 1$, we apply Karatsuba recursively up to obtain polynomials of size $2 \cdot 3^j$ and then we apply the strategy presented above to multiply such polynomials. The resulting complexity shows that, for $j \geq 1$ in $n = 2^i \times 3^j$, our approach yields better space and time complexities for the considered fields. The fact that our space complexity is better is due to the terms $-9.67n\log_3(n))$ in $\mathcal{C}$ and $\mathcal{C}'$ in (11) which are non-negligible for the above-mentioned sizes of polynomials.

**Table 6.** Complexities for polynomial multiplication of size $n = 2^i \cdot 3^j \in [160, 500]$

| Method | $162 = 2 \cdot 3^4$ | | | $192 = 2^6 \cdot 3$ | | | $216 = 2^3 \cdot 3^3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | #AND | #XOR | Del. | #AND | #XOR | Del. | #AND | #XOR | Del. |
| Karat. and Bern. [1] | 12147 | 30036 | 155 | 15309 | 35472 | 29 | 20655 | 50397 | 72 |
| Karat. and $\mathcal{C}$ in (11) | 4757 | 26217 | 43 | 7533 | 30126 | 28 | 8271 | 42765 | 39 |
| Karat. and $\mathcal{C}'$ in (11) | 3588 | 27386 | 43 | 5832 | 31827 | 28 | 6264 | 44772 | 39 |

| Method | $243 = 3^5$ | | | $256 = 2^8$ | | | $288 = 2^5 \cdot 3^2$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | #AND | #XOR | Del. | #AND | #XOR | Del. | #AND | #XOR | Del. |
| Karat. and Bern. [1] | 20901 | 52591 | 403 | 6561 | 34295 | 24 | 33291 | 79026 | 43 |
| Karat. and $\mathcal{C}$ in (11) | 11771 | 65167 | 50 | 6561 | 34295 | 24 | 14013 | 65661 | 35 |
| Karat. and $\mathcal{C}'$ in (11) | 8889 | 68049 | 50 | 6561 | 34295 | 24 | 10692 | 68982 | 35 |

| Method | $324 = 2^2 \cdot 3^4$ | | | $384 = 2^7 \cdot 3^1$ | | | $432 = 2^4 \cdot 3^3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | #AND | #XOR | Del. | #AND | #XOR | Del. | #AND | #XOR | Del. |
| Karat. and Bern. [1] | 36441 | 91239 | 158 | 45927 | 107757 | 32 | 61965 | 152700 | 75 |
| Karat. and $\mathcal{C}$ in (11) | 14271 | 79782 | 46 | 22599 | 91719 | 31 | 24813 | 129804 | 42 |
| Karat. and $\mathcal{C}'$ in (11) | 10764 | 83289 | 46 | 17496 | 96822 | 31 | 18792 | 135825 | 42 |

**Acknowledgement.** This work was supported in part by an NSERC grant awarded to Dr. Hasan.

## References

1. Bernstein, D.J.: Batch Binary Edwards. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 317–336. Springer, Heidelberg (2009)
2. Cenk, M., Koç, Ç., Özbudak, F.: Polynomial Multiplication over Finite Fields Using Field Extensions and Interpolation. In: 19th IEEE Symposium on Computer Arithmetic, ARITH 2009, pp. 84–91 (2009)
3. ElGamal, T.: A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
4. Fan, H., Hasan, M.A.: A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields. IEEE Transactions on Computers 56(2), 224–233 (2007)
5. Fan, H., Sun, J., Gu, M., Lam, K.-Y.: Overlap-free Karatsuba-Ofman Polynomial Multiplication Algorithm (May 2007)
6. Karatsuba, A.A.: The Complexity of Computations. In: Proceedings of the Steklov Institute of Mathematics, vol. 211, pp. 169–183 (1995)
7. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
8. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)

9. Miller, V.: Use of Elliptic Curves in Cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
10. Sunar, B.: A generalized method for constructing subquadratic complexity GF($2^k$) multipliers. IEEE Transactions on Computers 53, 1097–1105 (2004)
11. Toom, A.L.: The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers. Soviet Mathematics 3, 714–716 (1963)
12. Winograd, S.: Arithmetic Complexity of Computations. Society For Industrial & Applied Mathematics, U.S. (1980)

## A Lemmas and Their Proofs

In this section we provide two lemmas which gives the non-recursive solution to inductive expression. These solutions are required to obtain a non-recursive expression of the complexity of the formula presented in the paper. The proof of Lemma 1 can be found in [4].

**Lemma 1.** *Let $a, b$ and $i$ be positive integers and assume that $a \neq b$. Let $n = b^i$, $a \neq b$ and $a \neq 1$. The solution to the inductive relation* $\begin{cases} r_1 = e, \\ r_n = ar_{n/b} + cn + d, \end{cases}$ *is as follows*

$$r_n = (e + \frac{bc}{a-b} + \frac{d}{a-1})n^{\log_b(a)} - \frac{bc}{a-b}n - \frac{d}{a-1}. \tag{14}$$

**Lemma 2.** *Let $a, b$ and $i$ be positive integers. Let $n = b^i$ and $a = b$ and $a \neq 1$. The solution to the inductive relation* $\begin{cases} r_1 = e, \\ r_n = ar_{n/b} + cn + fn^\delta + d, \end{cases}$ *is*

$$r_n = n\left(e + \frac{fb^\delta}{a-b^\delta} + \frac{d}{a-1}\right) - n^\delta\left(\frac{fb^\delta}{a-b^\delta}\right) + cn\log_b(n) - \frac{d}{a-1}. \tag{15}$$

We prove the statement of Lemma 2 by induction on $i$ where $n = b^i$.

- For $i = 1$, i.e., $n = b$ we have

$$r_b = ar_1 + fb^\delta + cb + d = ae + fb^\delta + cb + d \tag{16}$$

  Now we compare this value of $r_b$ to the value given by the formula (15)

$$\begin{aligned} r_b &= ae + \tfrac{fb^\delta(b^\delta - a)}{b^\delta - a} + cb\log_b(b) + \tfrac{d(a-1)}{a-1} \\ &= ae + fb^\delta + bc + d. \end{aligned}$$

  Consequently, the formula in (15) is correct for $n = b$.
- We assume now that the formula is true for $i$ and we prove its correctness for $i + 1$. We first write

$$r_{b^{i+1}} = ar_{b^i} + fb^{i\delta} + cb^i + d$$

We then use the expression of $r_{b^i}$ given by the induction hypothesis

$$\begin{aligned} r_{b^{i+1}} &= a\left(a^i e + \tfrac{fb^\delta(b^{\delta i}-a^i)}{b^\delta - a} + cb^i + \tfrac{d(a^i-1)}{a-1})\right) + fb^{(i+1)\delta} + cb^{i+1} + d \\ &= a^{i+1}e + fb^\delta\left(\tfrac{ab^{\delta i}-a^{i+1}}{b^\delta-a} + b^{\delta i}\right) + c\left(iab^i + b^{i+1}\right) + d\left(\tfrac{a^{i+1}-a}{a-1} + 1\right) \\ &= a^{i+1}e + fb^\delta\left(\tfrac{ab^{\delta(i+1)}-a^{i+1}}{b^\delta-a}\right) + cb^{i+1}\log_b(b^{i+1}) + d\left(\tfrac{a^{i+1}-1}{a-1}\right) \end{aligned}$$

as required.

## B Bernstein's Three-Way Split Formula

In Table 7 we give the multi-evaluation and the products for Bernstein's formula.

**Table 7.** Cost of multi-evaluation and products for Bernstein's 3-way split formula

| Operations | Computations | Cost | |
|---|---|---|---|
| | | $\#\oplus$ | $\#\otimes$ |
| Multi-eval. | $R_1 = A_0 + A_1 + A_2$, $R'_1 = B_0 + B_1 + B_2$ | $4n/3$ | 0 |
| | $R_2 = A_1X + A_2X^2$, $R'_2 = B_1X + B_2X^2$ | $2n/3-2$ | 0 |
| | $R_3 = A_0 + R_2$, $R'_3 = B_0 + R'_2$ | $2n/3-2$ | 0 |
| | $R_4 = R_1 + R_2$, $R'_4 = R'_1 + R'_2$ | $2n/3-2$ | 0 |
| Products | $P_0 = A_0B_0$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| | $P_1 = R_1R'_1$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| | $P_2 = R_3R'_3$ | $S_\oplus(n/3+2)$ | $S_\otimes(n/3+2)$ |
| | $P_3 = R_4R'_4$ | $S_\oplus(n/3+2)$ | $S_\otimes(n/3+2)$ |
| | $P_4 = A_2B_2$ | $S_\oplus(n/3)$ | $S_\otimes(n/3)$ |
| Total | | $3S_\oplus(\frac{n}{3}) + 2S_\oplus(\frac{n}{3}+2)$ $+10n/3-6$ | $3S_\otimes(\frac{n}{3})$ $+2S_\otimes(\frac{n}{3}+2)$ |

In Table 8 we give explicit computations for the reconstruction of Bernstein's formula.

**Table 8.** Cost of reconstruction for Bernstein's 3-way split formula

| | Computations | $\#\oplus$ |
|---|---|---|
| Reconstruction | $S = P_2 + P_3$, | $2n/3+1$ |
| | $U = P_0 + (P_0 + P_1)X^{n/3}$ | $n-2$ |
| | $V = P_2 + S(X^{n/3} + X)$ | $n+4$ |
| | $W = U + V + P_4(X^4 + X)$ | $7n/3-3$ |
| | $W' = W/(X^2 + X)$ | $n-2$ |
| | $W'' = W(X^{2n/3} + X^{n/3})$ | $2n/3-1$ |
| | $C = U + P_4(X^{4n/3} + X^{n/3}) + W''$ | $5n/3-3$ |
| Total | | $25n/3-6$ |

Let us clarify the computation of $S$ in Table 8: the coefficients of $X^{2n/3+2}$ and $X^{2n/3+1}$ in $P_2$ are the same as the coefficients of the corresponding terms in $P_3$, therefore the degree of $P_2 + P_3$ is of degree $2n/3$ and this requires only $2n/3+1$ bit additions.