Improved Three-Way Split Formulas for Binary Polynomial and Toeplitz Matrix Vector Products

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

[1]ECE Department and CACR, University of Waterloo, Waterloo, Ontario, Canada.

[2]Team DALI, Université de Perpignan, Perpignan, France.

[3]LIRMM, Université Montpellier 2, Montpellier, France.

Abstract

In this paper we consider 3-way split formulas for binary field multiplication and Toeplitz matrix vector product. We first recall the best known formulas and derive the complexity of parallel multipliers based on these formulas. We then propose a new set of 3-way split formulas. We evaluate their complexities and provide a comparison.

1 Introduction

Several cryptographic applications like those relying on elliptic curve cryptography [10], [8] or Galois Counter Mode [9] require efficient finite field arithmetic. For example, ciphering a message using the ElGamal scheme [4] 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 of degree n. 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 [7] or Toom-Cook [12], [13] are considered to be most appropriate. Several parallel multipliers have been proposed based on such

approaches [11], [6]. 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 mostly *logarithmic* in time. When such a subquadratic complexity multiplier is implemented in hardware in bit parallel fashion, well known approaches include 2-way split formula with three recursive multiplications and 3-way split formula with six recursive multiplications [11], [6]. Recently, Bernstein in [1] has proposed a 3-way split formula with only *five* recursive multiplications.

The multiplication in a finite field can be also expressed as Toeplitz matrix vector product (TMVP) [5]. Furthermore, the authors of [5] have shown that it is also possible to design subquadratic space complexity algorithm for TMVP. For this purpose, they use a two-way split formula which expresses a TMVP of size n in terms of three TMVPs of size $n/2$ and a three-way formula which expresses a TMVP of size n in terms of six TMVPs of size $n/3$.

In this paper we also consider 3-way splits and propose new formulas for binary polynomial multiplication and Toeplitz matrix vector product. We first present a new 3-way formula with six recursive multiplication which improves the number of bit operations of the previous known formulas of [6], [11]. We then present a formula 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 the interpolation methods. This leads to a Toom-Cook like formula. Using the method of Winograd [13], we transform the 3-way split formula for polynomial multiplication to a formula for Toeplitz matrix vector product. In each case we study the recursive complexity of the proposed formula and we derive 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 a 3-way split formula for polynomial multiplication. We then review Sunar's and Bernstein's formulas and evaluate a non-recursive form of their complexities. In Section 3 we present a new set of 3-way formulas with six recursive multiplications for polynomial multiplication. In Section 4 we present a new set of 3-way formulas with five recursive multiplication based on field extension for polynomial multiplication. In Section 5, we extend the latter formulas to Toeplitz matrix vector product. Complexity comparison of these approaches is discussed in Section 6, and further complexity results are presented for some less constrained n in Section 7. We end this paper with some concluding remarks in Section 8.

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 two specific 3-way split formulas– the first one requires six recursive multiplications [11] and the other needs five [1]. We also study complexity results of both formulas.

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: $A = A_0 + A_1 X^{n/3} + A_2 X^{2n/3}$ and $B = B_0 + B_1 X^{n/3} + B_2 X^{2n/3}$, 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 resulting in $A = A_0 + A_1 Y + A_2 Y^2$ and $B = B_0 + B_1 Y + B_2 Y^2$. Thus, each of A and B is a degree two polynomial in Y. The product C of A and B is then a polynomial in Y of degree four (i.e. it has five terms). Polynomial C can be determined if its values are known at five distinct points. To this end, we fix four elements $\alpha_1, \ldots, \alpha_4 \in \mathcal{R}$ plus the infinity element $\alpha_5 = \infty$. Then, 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 provide the evaluations $C(\alpha_i)$ of $C(Y) = A(Y) \times B(Y)$ at these 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

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

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

2.2 Review of 3-way splitting formula with six multiplications

Let $A(X)$ and $B(X)$ be two binary polynomials of degree $n-1$ in $\mathbb{F}_2[X]$, where n is a power of 3. As before we split these two polynomials in three parts and obtain their corresponding degree 2 polynomials $A(Y) = A_0 + A_1 Y + A_2 Y^2$ and $B(Y) = B_0 + B_1 Y + B_2 Y^2$ in the indeterminate Y. There are only 2 elements in $\mathbb{F}_2$; consequently there are not enough elements to apply the method of the previous subsection. To overcome to this problem, Winograd in [13], [11] proposed to replace the two missing products $A(\alpha_3) \times B(\alpha_3)$ and $A(\alpha_4) \times B(\alpha_4)$ by a multiplication modulo the degree 2 polynomial Y^2+Y+1. Indeed, the multi-evaluation and interpolation approach can be seen as a special case of the Chinese Remainder Theorem (CRT): it consists of the multiplication of two polynomials modulo $\prod_{i=1}^{5}(Y-\alpha_i)$ by computing the product C separately modulo $Y-\alpha_i$ and the reconstruction of C using a Lagrange-like interpolation. Winograd proposed to perform the multiplication modulo $Y(Y-1)(Y^2+Y+1)(Y-\infty)$

$$\begin{aligned}
C(0) &= A(0)B(0) = A_0 B_0, \\
C(1) &= A(1)B(1) = (A_0 + A_1 + A_2)(B_0 + B_1 + B_2), \\
C(Y) \mod (Y^2+Y+1) &= (A_0 + A_2 + (A_1 + A_2)Y)(B_0 + B_2 + (B_1 + B_2)Y) \mod (Y^2+Y+1), \\
C(\infty) &= A(\infty)B(\infty) = A_2 B_2.
\end{aligned}$$

The multiplication modulo $(Y^2 + Y + 1)$ consists of a product of degree one polynomial. To perform this product Winograd used the Karatsuba formula, which requires 3 multiplications $(A_0 + A_2)(B_0 + B_2)$ and $(A_0 + A_1)(B_0 + B_1)$ and $(A_1 + A_2)(B_1 + B_2)$. The product C is reconstructed using the Chinese Remainder Theorem. In Table 1 we have reported the formula of [11] which is a modified version of this Winograd's 3-way split formula. Indeed, in Table 1 the three terms P_3, P_4 and P_5 are the multiplications corresponding to the product modulo $Y^2 + Y + 1$. Also note that $A(1)B(1)$ has been modified to save some computation, since we can write

$$(A_0 + A_1 + A_2)(B_0 + B_1 + B_2) = P_1 + P_2 + P_5 + P_3 + P_0 + P_4,$$

the product $P_1 = A_1B_1$ can be used in place of $A(1)B(1)$.

TABLE 1

3-way split formula with six multiplications of [11]

Operations	Underlying computations	Cost	
		$\#\oplus$	$\#\otimes$
Prod.	$P_0 = A_0B_0$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_1 = A_1B_1$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_2 = A_2B_2$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_3 = (A_1 + A_2)(B_1 + B_2)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
	$P_4 = (A_0 + A_1)(B_0 + B_1)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
	$P_5 = (A_0 + A_2)(B_0 + B_2)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
Reconst.	$R_0 = P_0 + P_1$	$2n/3 - 1$	0
	$R_1 = P_4 + R_0$	$2n/3 - 1$	0
	$R_2 = P_5 + R_0 + P_2$	$4n/3 - 2$	0
	$R_3 = P_3 + P_1 + P_2$	$4n/3 - 2$	0
	$C = P_0 + R_0X^{n/3} + R_1X^{2n/3} + R_2X^{3n/3} + P_2X^{4n/3}$	$4n/3 - 4$	0
Total		$6\mathcal{S}_\oplus(n/3) + 22n/3 - 10$	$6\mathcal{S}_\otimes(n/3)$

We have also reported in Table 1 the cost in bit addition ($\oplus$) and bit multiplication ($\otimes$) of each step of the 3-way split formula. The resulting overall cost, in terms of bit addition $S_\oplus(n)$ and bit multiplication $S_\otimes(n)$, has the recursive form given below

$$\begin{cases} S_\oplus(n) &= 6\mathcal{S}_\oplus(n/3) + 22n/3 - 10, \\ S_\otimes(n) &= 6\mathcal{S}_\otimes(n/3). \end{cases}$$

The following lemma gives us a general solution to such a recursive equation. This will help us to obtain the non-recursive expression of $S_\oplus$ and $S_\otimes$ in terms of n.

Lemma 1 ([5]): 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*

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

We apply (1) to our situation with the initial condition $\mathcal{S}_\oplus(1) = 0$ and $\mathcal{S}_\otimes(1) = 1$ and we obtain

$$\begin{cases} \mathcal{S}_\oplus(n) &= \frac{16}{3}n^{\log_3(6)} - 22\frac{n}{3} + 2 \\ \mathcal{S}_\otimes(n) &= n^{\log_3(6)} \end{cases} \tag{2}$$

Now we assume that the computations are performed in parallel. A careful study of the formula of Table 1 yields that the delay of the recursive products step is equal to $2D_\oplus + \mathcal{D}(n/3)$, where $D_\oplus$ represents the delay of a bit addition and $\mathcal{D}(n/3)$ the delay of the product of two degree $n/3$ polynomials. The reconstruction step requires $2D_\oplus$; consequently, adding these two delays results in

$$\mathcal{D}(n) = 4D_\oplus + \mathcal{D}(n/3).$$

Since we know that $\mathcal{D}(0) = D_\otimes$, we can easily transform this recursive expression, to a non-recursive form $\mathcal{D}(n) = 4\log_3(n)D_\oplus + D_\otimes$.

2.3 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 $\mathbb{F}_2[X]$. 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 $\mathcal{R} = \mathbb{F}_2[X]$

$$A = A_0 + A_1Y + A_2Y^2 \text{ and } B = B_0 + B_1Y + B_2Y^2 \text{ with } \deg_X A_i, \deg_X B_i < n/3.$$

Bernstein uses a multi-evaluation and interpolation approach by evaluating the polynomials at these five elements $1, 0, X, X+1$ and ∞ 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 ∞ as follows

$$\begin{aligned} P_0 &= A_0B_0 \quad \text{(eval. at 0)}, \\ P_1 &= (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) \quad \text{(eval. at 1)}, \\ P_2 &= (A_0 + A_1X + A_2X^2)(B_0 + B_1X + B_2X^2) \quad \text{(eval. at } X\text{)}, \\ P_3 &= \big((A_0 + A_1 + A_2) + (A_1X + A_2X^2)\big) \\ &\qquad \times \big((B_0 + B_1 + B_2) + (B_1X + B_2X^2)\big) \quad \text{(eval. at } X+1\text{)}, \\ P_4 &= A_2B_2 \quad \text{(eval. at } \infty\text{)}. \end{aligned}$$

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

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

For Bernstein's approach we give the explicit computations algorithm along with their cost in the remainder of this section.

Explicit computations of the multi-evaluation and recursive products.

In Table 2 we give the explicit computation of the multi-evaluation and the products for Bernstein's approach.

TABLE 2

Cost of multi-evaluation and products for Bernstein's 3-way split formula

Operations	Underlying 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			$3S_\oplus(\frac{n}{3}) + 2S_\oplus(\frac{n}{3}+2)$ $+10n/3-6$	$3S_\otimes(\frac{n}{3})$ $+2S_\otimes(\frac{n}{3}+2)$

The complexities of the computation of the five products P_0, P_1, P_2, P_3 and P_4 are given in Table 2. 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 degrees equal to $(2n/3-2)$ and the degrees of P_2 and P_3 are equal to $(2n/3+2)$.

The formula in Table 2 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 $P = \sum_{i=0}^{n/3+1} p_iX^i$ and $Q = \sum_{i=0}^{n/3+1} q_iX^i$. 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

$$\begin{aligned} PQ \;=\; & \underbrace{P'Q'}_{M_1} + \underbrace{(p_{n/3}X^{n/3} + p_{n/3+1}X^{n/3+1})Q'}_{M_2} + \underbrace{(q_{n/3}X^{n/3} + q_{n/3+1}X^{n/3+1})P'}_{M_3} \\ & + \underbrace{(p_{n/3}X^{n/3} + p_{n/3+1}X^{n/3+1})(q_{n/3}X^{n/3} + q_{n/3+1}X^{n/3+1})}_{M_4}. \end{aligned} \tag{4}$$

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 costly, 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 addition 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:

$$\begin{cases} \mathcal{S}_{\oplus}(n/3+2) = \mathcal{S}_{\oplus}(n/3) + 4n/3, \\ \mathcal{S}_{\otimes}(n/3+2) = \mathcal{S}_{\otimes}(n/3) + 4n/3 + 4. \end{cases} \tag{5}$$

Explicit computations of the reconstruction.

We now review the sequence of computations proposed by Bernstein in [1] for the reconstruction. This sequence first computes the two polynomials U and V defined in (3) and then computes C. The details are given in Table 3.

TABLE 3

Cost of reconstruction for Bernstein's 3-way split formula

Operations	Underlying 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		$25n/3-6$

Note that polynomial P_0, P_1 and P_4 are of degree $2n/3-2$ and P_2 and P_3 have degree $2n/3+2$. We note that, for the computation of S in Table 3, the coefficients of $X^{2n/3+2}$ and $X^{2n/3+1}$ of P_2 are the same as the coefficients of the corresponding terms of P_3; therefore the degree of $P_2 + P_3$ is $2n/3$ and this requires only $2n/3+1$ bit additions.

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 (3) (i.e., the computation of W' in Table 3), 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 (Tables 2 and 3). By adding the number of bit additions listed in the two tables 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 (5). We then obtain the following:

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

Now we apply Lemma 1 and we obtain

$$\mathcal{C} = \begin{cases} \mathcal{S}_\oplus(n) &= \frac{37}{2} n^{\log_3(5)} - \frac{43n}{2} + 3, \\ \mathcal{S}_\otimes(n) &= 7n^{\log_3(5)} - 4n - 2. \end{cases} \tag{6}$$

Delay of parallel computation based on Bernstein's method.

Here we evaluate the delay of a parallel multiplier based on Bernstein's formula. For this, we have drawn a data-flow graph of the multi-evaluation and the reconstruction part of the computation. The graph is 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 $\mathcal{D}(n/3) + D_\oplus$ (see (4)), 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 the expression for the critical path $\mathcal{D}(n) = \mathcal{D}(n/3) + (n+8)D_\oplus$ into a non-recursive form, by applying it recursively and using $\mathcal{D}(1) = D_\otimes$ as shown below

$$\begin{aligned} \mathcal{D}(n) &= (n+8)D_\oplus + (n/3+8)D_\oplus + (n/9+8)D_\oplus + \ldots + (3+8)D_\oplus + D_\otimes \\ &= (8\log_3(n) + \tfrac{3n}{2} - \tfrac{3}{2})D_\oplus + D_\otimes. \end{aligned} \tag{7}$$

3 IMPROVEMENT OF 3-WAY SPLIT FORMULA WITH SIX RECURSIVE MULTIPLICATIONS

In this section we present a new 3-way formula with six recursive multiplications which reduces the complexity of the reconstruction computations. We have followed the idea of Bernstein in [1] which saves some computation of the Karatsuba formula by re-arranging the reconstruction process. Here we improve the reconstruction formula of Table 1.

As before, let A and B be two degree $n-1$ polynomials in $\mathbb{F}_2[X]$, where n is a power of 3. We split A and B in three parts $A = A_0 + A_1 X^{n/3} + A_2 X^{2n/3}$ and $B = B_0 + B_1 X^{n/3} + B_2 X^{2n/3}$, where A_i and B_i

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

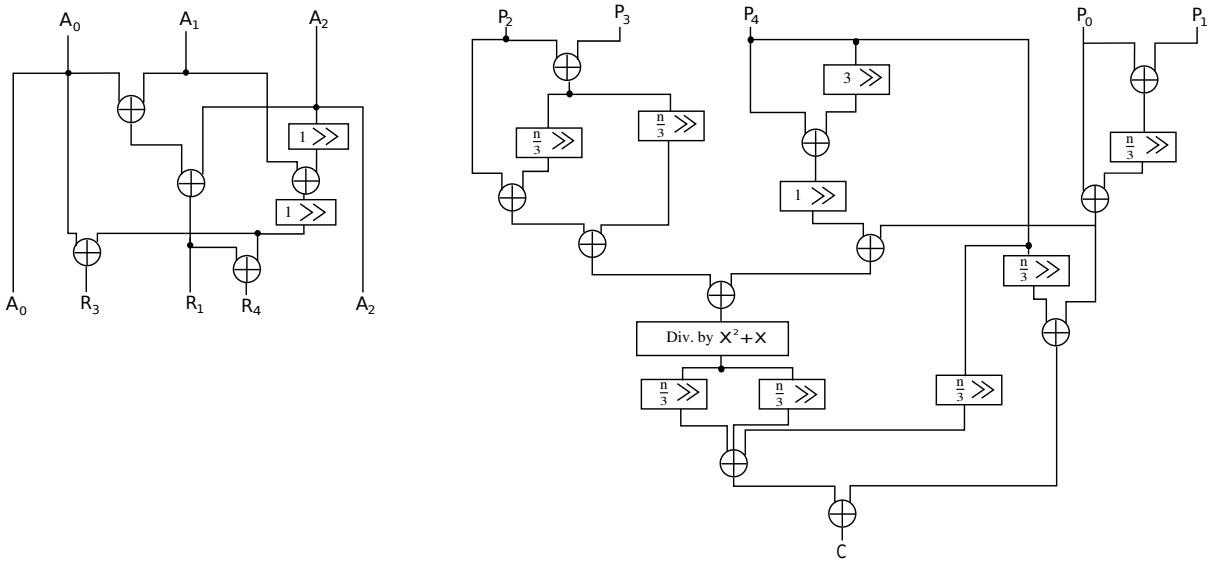


TABLE 4

Proposed 3-way split formula with six multiplications

Operations	Underlying computations	Cost	
		$\#\oplus$	$\#\otimes$
Rec. Prod.	$P_0 = A_0B_0$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_1 = A_1B_1$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_2 = A_2B_2$	$S_\oplus(n/3)$	$S_\otimes(n/3)$
	$P_3 = (A_1 + A_2)(B_1 + B_2)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
	$P_4 = (A_0 + A_1)(B_0 + B_1)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
	$P_5 = (A_0 + A_2)(B_0 + B_2)$	$S_\oplus(n/3) + 2n/3$	$S_\otimes(n/3)$
Reconst.	$R_0 = (P_0 + X^{n/3}P_1 + X^{2n/3}P_2)$	$(2n/3 - 2)$	0
	$R_1 = R_0(1 + X^{n/3} + X^{2n/3})$	$(2n - 2)$	0
	$C = R_1 + P_4X^{n/3} + P_5X^{2n/3} + P_3X^{3n/3}$	$(2n - 3)$	0
Total		$6\mathcal{S}_\oplus(n/3) + 20n/3 - 7$ $= \frac{79}{15}n^{\log_3(6)} - 20n/3 + 7/5$	$6\mathcal{S}_\otimes(n/3)$ $= n^{\log_3(6)}$

have degree $n/3-1$. We perform the multiplication of A and B using the six recursive multiplications of Table 1. The product C is reconstructed in Table 1 in terms of six products using the following expression

$$C = P_0 + (P_4 + P_0 + P_1)X^{n/3} + (P_5 + P_0 + P_1 + P_2)X^{2n/3} + (P_3 + P_1 + P_2)X^{3n/3} + P_2X^{4n/3}.$$

Now we rewrite this expression of C as follow

$$C = (P_0 + X^{n/3}P_1 + X^{2n/3}P_2)(1 + X^{n/3} + X^{2n/3}) + P_4X^{n/3} + P_5X^{2n/3} + P_3X^{3n/3}$$

Then we define $R_0 = (P_0 + X^{n/3}P_1 + X^{2n/3}P_2),\ R_1 = R_0(1 + X^{n/3} + X^{2n/3})$. We now obtain the following expression of C in terms of P_i and R_i

$$C = R_1 + P_4X^{n/3} + P_5X^{2n/3} + P_3X^{3n/3}.$$

Complexity evaluation. The computation of R_0 and R_1 requires $(2n/3 - 2)$ and $(2n - 2)$ additions respectively. For the computation of C we need $(2n - 3)$ additions. The resulting formula is summarized in Table 4.

We save $2n/3$ bit additions compared to the formula given in Table 1. It is thus asymptotically slightly more efficient than the previously known 3-way split formula with six recursive multiplications.

Concerning the delay, by analyzing the formula of Table 4, we see that the critical path delay is $4D_{\oplus} + \mathcal{D}(n/3)$. Consequently, the overall delay of the multiplication when the proposed formula is applied recursively is equal to $4\log_3(n)D_{\oplus} + D_{\otimes}$.

4 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 $A = \sum_{i=0}^{n-1} a_iX^i$ and $B = \sum_{i=0}^{n-1} b_iX^i$ 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 face 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 [13], [2] 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, α, $\alpha+1$ and ∞. The resulting evaluations and recursive multiplication are given below:

$$\begin{array}{rcll} P_0 & = & A_0B_0 & \text{in } \mathbb{F}_2[X], \\ P_1 & = & (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) & \text{in } \mathbb{F}_2[X], \\ P_2 & = & (A_0 + A_2 + \alpha(A_1 + A_2))(B_0 + B_2 + \alpha(B_1 + B_2)), & \text{in } \mathbb{F}_4[X], \\ P_3 & = & (A_0 + A_1 + \alpha(A_1 + A_2))(B_0 + B_1 + \alpha(B_1 + B_2)), & \text{in } \mathbb{F}_4[X], \\ P_4 & = & A_2B_2 & \text{in } \mathbb{F}_2[X]. \end{array} \quad (8)$$

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

$$\begin{aligned} C = &(P_0 + X^{n/3}P_4)(1 + X^n) + (P_1 + (1+\alpha)(P_2 + P_3))(X^{n/3} + X^{2n/3} + X^n) \\ &+\alpha(P_2 + P_3)X^n + P_2X^{2n/3} + P_3X^{n/3} \end{aligned} \tag{9}$$

Note that if we evaluate a binary polynomial at $0, 1$ or ∞ we obtain a polynomial in $\mathbb{F}_2[X]$ and on the other hand if we evaluate the same polynomial at α 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 (8) and (9) 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 $\mathbb{F}_2[X]$.

4.1 Explicit 3-way splitting formulas

In this subsection, we provide a sequence of computations for (8) and (9) when A and B are taken in $\mathbb{F}_4[X]$ or in $\mathbb{F}_2[X]$. We split the computations of (8) and (9) in the three different steps: we first give the formulas for the multi-evaluation, then for the products and finally for the reconstruction.

TABLE 5

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		$22n/3$	$8n/3$

Multi-evaluation formulas. The proposed steps to compute the multi-evaluation of A and B are detailed in Table 5. 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 $(a_0 + b_1\alpha) + (b_0 + b_1\alpha) = (a_0 + b_0) + (a_1 + b_1)\alpha$ and requires 2 bit additions. Consequently, the sum of two degree $d - 1$ polynomials in $\mathbb{F}_4[X]$ requires $2d$ bit additions.

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

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 can evaluate each step of Table 5 and then deduce the complexity of the multi-evaluation by adding the cost of each step.

TABLE 6

Cost of products for the new three-way split formulas

Computations	Cost in $\mathbb{F}_4$		Cost in $\mathbb{F}_2$	
	$\#\oplus$	$\#\otimes$	$\#\oplus$	$\#\otimes$
$P_0 = A_0B_0$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
$P_1 = R_6R_6'$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
$P_2 = R_5R_5'$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$
$P_3 = R_4R_4'$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$
$P_4 = A_2B_2$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
Total	$5S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$	$5S_{\mathbb{F}_4,\otimes}(\frac{n}{3})$	$3S_{\mathbb{F}_2,\oplus}(\frac{n}{3}) + 2S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$	$3S_{\mathbb{F}_2,\otimes}(\frac{n}{3}) + 2S_{\mathbb{F}_4,\oplus}(\frac{n}{3})$

Recursive products. In Table 6 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 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 7 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 do not need to add their bits corresponding to α.
- The polynomial C is in $\mathbb{F}_2[X]$; consequently, we do not need to compute the coefficients corresponding to α. Indeed, if $a = a_0 + a_1\alpha$ and $b = b_0 + b_1\alpha$, we denote $[a+b]_{const} = a_0 + b_0$ which requires only one bit addition. We use the same notation for the polynomials.

TABLE 7

Three-way split formulas - Reconstruction

Reconstruction in $\mathbb{F}_4$	
Underlying 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
$C = U_7 + U_5 + X^n U_2$ $+P_2X^{2n/3} + P_3X^{n/3}$	$20n/3-10$
Total	$36n/3-21$

Reconstruction in $\mathbb{F}_2$	
Underlying computations	$\#\oplus$
$U_1 = P_2 + P_3$	$4n/3-2$
$U_2 = [\alpha U_1]_{const}$	0
$U_3 = [(1+\alpha)U_1]_{const}$	$2n/3-1$
$U_4 = [P_1 + U_3]_{const}$	$2n/3-1$
$U_5 = [U_4(X^{n/3} + X^{2n/3} + X^{3n/3})]_{const}$	$2n/3-2$
$U_6 = [P_0 + X^{n/3}P_4]_{const}$	$n/3-1$
$U_7 = [U_6(1+X^n)]_{const}$	0
$C = [U_7 + U_5 + X^n U_2]_{const}$ $+P_2X^{2n/3} + P_3X^{n/3}]_{const}$	$10n/3-5$
Total	$21n/3-12$

4.2 Complexity evaluation

We now evaluate the complexity of the formulas given in Tables 5, 6 and 7. 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

$$\begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) &= 5\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3) + 58n/3 - 21, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) &= 5\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3). \end{cases} \tag{10}$$

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 between 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 (10) 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 (10) 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.

$$\mathcal{C} = \begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) &= \frac{107}{4}n^{\log_3(5)} - 29n + \frac{21}{4}, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) &= 4n^{\log_3(5)}. \end{cases} \quad \mathcal{C}' = \begin{cases} \mathcal{S}_{\mathbb{F}_4,\oplus}(n) &= \frac{111}{4}n^{\log_3(5)} - 29n + \frac{21}{4}, \\ \mathcal{S}_{\mathbb{F}_4,\otimes}(n) &= 3n^{\log_3(5)}. \end{cases} \tag{11}$$

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 5, 6 and 7, we obtain the number of bit additions (resp. bit multiplications) expressed in terms of $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ and $\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$ (resp. $\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$ and $\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$) as follows

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

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 (11). 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 (12), we replace $\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ by its expression given in (11) 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: $\mathcal{S}_{\mathbb{F}_2,\oplus}(n) = \frac{107}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) - \frac{55n}{2} + \frac{3}{4}$.

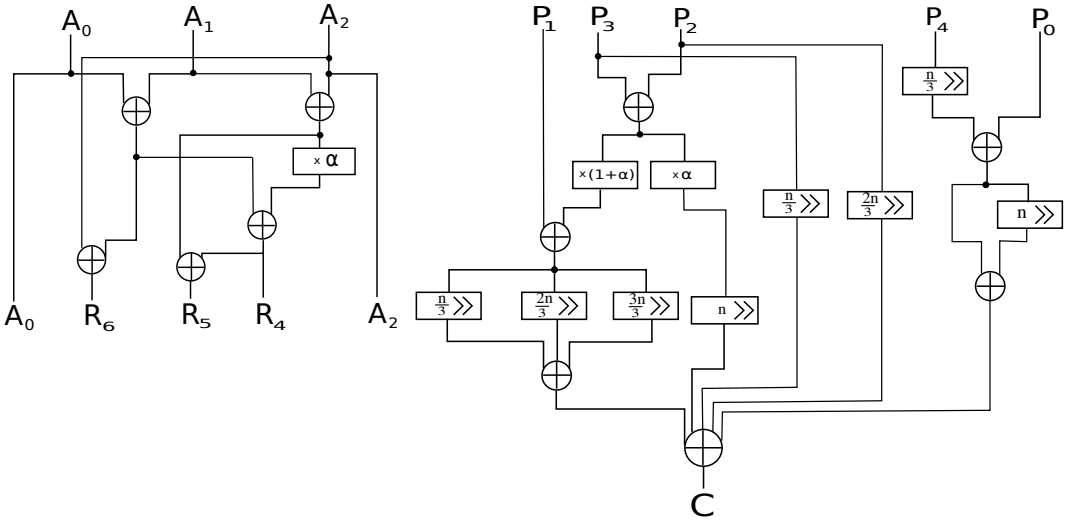
We then apply the same method to other complexities. Below we give the final non-recursive expression for each case.

$$\mathcal{C} = \begin{cases} \mathcal{S}_{\mathbb{F}_2,\oplus}(n) &= \frac{107}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) \\ & \quad -\frac{55n}{2} + \frac{3}{4} \\ \mathcal{S}_{\mathbb{F}_2,\otimes}(n) &= 4n^{\log_3(5)} - 3n \end{cases} \quad \mathcal{C}' = \begin{cases} \mathcal{S}_{\mathbb{F}_2,\oplus}(n) &= \frac{111}{4}n^{\log_3(5)} - \frac{29}{3}n\log_3(n) \\ & \quad -\frac{57n}{2} + \frac{3}{4} \\ \mathcal{S}_{\mathbb{F}_2,\otimes}(n) &= 3n^{\log_3(5)} - 2n \end{cases} \tag{13}$$

4.3 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 the operations for these two cases ($\mathbb{F}_4$ and $\mathbb{F}_2$) are essentially the same: their only difference is on the reconstruction: when then considered multiplication is in $\mathbb{F}_2[X]$ 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 α and then ends in R_1. Since a multiplication by α 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:

$$\mathcal{D}_{\mathbb{F}_4}(n) = (10\log_3(n) + 2)D_{\oplus} + D_{\otimes}. \tag{14}$$

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 α 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

$$\mathcal{D}_{\mathbb{F}_2}(n) = 10\log_3(n)D_{\oplus} + D_{\otimes}. \tag{15}$$

5 THREE-WAY FORMULAS FOR TOEPLITZ MATRIX VECTOR PRODUCT

The problem of multiplication of elements in $\mathbb{F}_{2^n}$ can be expressed as multiplication of binary polynomials of degree $< n$. Such polynomial multiplication has been the focus of the previous sections. The problem of multiplication of elements in $\mathbb{F}_{2^n}$ can alternatively be expressed as the product of an $n \times n$ Toeplitz matrix and a vector of size n (cf. [5]). In this section, we consider efficient computation of such matrix and vector product.

We recall that a matrix $T = [t_{i,j}]_{i,j=0,\ldots,n-1}$ is Toeplitz if its entries satisfy $t_{i,j} = t_{i-1,j-1}$ for $i, j = 1, \ldots, n-1$. We use the tensor technique (which can be found in [13]) which transforms a recursive formula for polynomial multiplication to a similar recursive formula for Toeplitz matrix vector product. We consider two degree 2 polynomials $A = A_0 + A_1Y + A_2Y^2$ and $B = B_0 + B_1Y + B_2Y^2$. If we equate the direct computation of $C = A \times B$ with the expression of C given in (9) and (8), we obtain the following five identities

$$\begin{aligned} A_0B_0 &= P_0, \\ A_0B_1 + A_1B_0 &= P_1 + P_2 + P_4 + \alpha(P_2 + P_3), \\ A_0B_2 + A_1B_1 + A_2B_0 &= P_1 + P_3 + \alpha(P_2 + P_3), \\ A_1B_2 + A_2B_1 &= P_0 + P_1 + P_2 + P_3, \\ A_2B_2 &= P_4. \end{aligned} \tag{16}$$

Referring to (16) we pre-multiply on the left the first equation (or line) by T_4, the second by T_3 and so on up to the last one which is multiplied by T_0. We sum the resulting five equations and sort the left side relative to B_i yielding

$$\begin{aligned} &(T_4A_0 + T_3A_1 + T_2A_2)B_0 + (T_3A_0 + T_2A_1 + T_1A_2)B_1 + (T_2A_0 + T_1A_1 + T_0A_2)B_2 \\ &= T_4P_0 + T_3(P_1 + P_2 + P_4 + \alpha(P_2 + P_3)) + T_2(P_1 + P_3 + \alpha(P_2 + P_3)) \\ &\quad + T_1(P_0 + P_1 + P_2 + P_3) + T_0P_4. \end{aligned} \tag{17}$$

Let us denote $f(T, A, B)$ as the expression on the left side of (17). Now we transform the right side of the previous expression. We expand the five products of the right side and sort the results relative to P_i to obtain

$$\begin{aligned} f(T, A, B) &= (T_4 + T_1)P_0 + (T_3 + T_2 + T_1)P_1 + (T_3(1 + \alpha) + \alpha T_2 + T_1)P_2 \\ &\quad + (\alpha T_3 + (1 + \alpha)T_2 + T_1)P_3 + (T_3 + T_0)P_4. \end{aligned} \tag{18}$$

Now we make the following transformations: referring to (8), note that each P_i has a form $L_i(A)L_i(B)$, where $L_i(A)$ and $L_i(B)$ are linear (over $\mathbb{F}_4$) in terms of A_i and B_i, respectively. Since on the right hand

side of (18), the coefficient of P_i has also a linear expression $L'_i(T)$ over $\mathbb{F}_4$, we can rewrite such terms as $L'_i(T)P_i = Q_iL_i(B)$, where $Q_i = L'_i(T)L_i(A)$, for $i = 0, 1, \cdots, 4$, are as follows:

$$\begin{aligned} Q_0 &= (T_4 + T_1)A_0, \\ Q_1 &= (T_3 + T_2 + T_1)(A_0 + A_1 + A_2), \\ Q_2 &= (T_3 + T_1 + \alpha(T_2 + T_3))(A_0 + A_2 + \alpha(A_1 + A_2)), \\ Q_3 &= (\alpha(T_3 + T_2) + T_2 + T_1)(A_0 + A_1 + \alpha(A_1 + A_2)), \\ Q_4 &= (T_3 + T_0)A_2. \end{aligned} \tag{19}$$

We replace each $L'_i(T)P_i$ by $Q_iL_i(B)$ on the right side of (18) and we expand the five linear terms $L_i(B)$. Then we sort the new expression in terms of B_i to obtain

$$\begin{aligned} f(T, A, B) &= Q_0B_0 + Q_1(B_0 + B_1 + B_2) + Q_2(B_0 + B_2 + \alpha(B_1 + B_2)) \\ &\quad + Q_3(B_0 + B_1 + \alpha(B_1 + B_2)) + Q_4B_2 \\ &= (Q_0 + Q_1 + Q_2 + Q_3)B_0 + (Q_1 + Q_3 + \alpha(Q_2 + Q_3))B_1 \\ &\quad + (Q_1 + Q_2 + Q_4 + \alpha(Q_2 + Q_3))B_2. \end{aligned} \tag{20}$$

Finally, going back to (17), we can equate the coefficients on B_i of the right side of (20) with the coefficient of B_i in

$$f(T, A, B) = (T_4A_0 + T_3A_1 + T_2A_2)B_0 + (T_3A_0 + T_2A_1 + T_1A_2)B_1 + (T_2A_0 + T_1A_1 + T_0A_2)B_2,$$

The resulting three equations in T_i and A_i are

$$\begin{aligned} T_4A_0 + T_3A_1 + T_2A_2 &= Q_0 + Q_1 + Q_2 + Q_3, \\ T_3A_0 + T_2A_1 + T_1A_2 &= Q_1 + Q_3 + \alpha(Q_2 + Q_3), \\ T_2A_0 + T_1A_1 + T_0A_2 &= Q_1 + Q_2 + Q_4 + \alpha(Q_2 + Q_3). \end{aligned}$$

The left side corresponds to the product of a 3×3 Toeplitz matrix T with a column vector A of size 3 and the right side gives the required 3-way split formula to compute the product, i.e.,

$$\begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 + Q_2 + Q_4 + \alpha(Q_2 + Q_3) \\ Q_1 + Q_3 + \alpha(Q_2 + Q_3) \\ Q_0 + Q_1 + Q_2 + Q_3 \end{bmatrix}.$$

In the sequel, we will apply this formula to a TMVP of size n and split it in three parts. In the formula, if T and A have their entries in $\mathbb{F}_2$, then the products Q_0, Q_1 and Q_4 are TMVP of size $n/3$ with entries in $\mathbb{F}_2$, while the products Q_2 and Q_3 have their entries in $\mathbb{F}_4$. Consequently, we also need to evaluate the complexity of the formula when the TMVP has its entries in $\mathbb{F}_4$ and then we can deduce the complexity for TMVP with entries in $\mathbb{F}_2$.

We provide explicit formulas and their complexity evaluation for the $\mathbb{F}_4$ and the $\mathbb{F}_2$ cases at the same time since most of the formulas are the same. We will split this study in three parts: the multi-evaluation, the pairwise multiplication and the reconstruction.

Multi-evaluation. In Table 8 we give a sequence of computations to perform the multi-evaluation on the matrix T. In that table, we also give the cost of each operation separated in two columns based on whether the entries of T_i are in $\mathbb{F}_4$ or $\mathbb{F}_2$. We use the following fact to compute the complexity of each step: first, a $d \times d$ Toeplitz matrix can be defined by $2d-1$ coefficients only and thus an addition of two such matrices with entries in $\mathbb{F}_4$ costs $2(2d-1)$ bit additions and secondly, an addition of two $d \times d$ Toeplitz matrices with entries in $\mathbb{F}_2$ requires $2d-1$ bit additions. In the special case where the entries are in $\mathbb{F}_2$, in Table 8 the multiplication by α in R_3 is free of computation. Similarly, since R_3 has all its entries in $\alpha\mathbb{F}_2$ the additions with a matrix with entries in $\mathbb{F}_2$ requires no additions.

TABLE 8

Multi-evaluation of T

Underlying computations	Entries in $\mathbb{F}_4$ $\#\oplus$	Entries in $\mathbb{F}_2$ $\#\oplus$
$R_1 = T_4 + T_1$,	$4n/3-2$	$2n/3-1$
$R_2 = T_3 + T_2$,	$4n/3-2$	$2n/3-1$
$R_3 = \alpha R_2 \ \ (= \alpha(T_3 + T_2))$	$2n/3-1$	0
$R_4 = T_1 + R_3 \ \ (= T_1 + \alpha(T_3 + T_2))$	$4n/3-2$	0
$R_5 = T_3 + R_4 \ \ (= T_3 + T_1 + \alpha(T_3 + T_2))$	$4n/3-2$	$2n/3-1$
$R_6 = T_2 + R_4 \ \ (= T_2 + T_1 + \alpha(T_3 + T_2))$	$4n/3-2$	$2n/3-1$
$R_7 = T_1 + R_2 \ \ (= T_1 + T_2 + T_3)$	$4n/3-2$	$2n/3-1$
$R_8 = T_3 + T_0$	$4n/3-2$	$2n/3-1$
Total	$10n-15$	$12n/3-6$

In Table 9 we give the sequence of computations for the multi-evaluation of the vector A. We also give the cost of each operation taking account that the entries are in $\mathbb{F}_2$ or $\mathbb{F}_4$. Specifically, we use the same tricks used for the multi-evaluation of T to save some computations when the entries are in $\mathbb{F}_2$.

Products. We provide in Table 10 the five products of entries of size $n/3$. Note that there are three TMVPs with matrices and vectors with entries in $\mathbb{F}_2$ which are less costly. This lowers the complexity when we apply our formulas recursively.

Reconstruction. We now give the sequence of computations for the reconstruction of the product $W = T \cdot A$ and we evaluate its cost. We give two sets of formulas which correspond to the situations where T and A have their entries in $\mathbb{F}_4$ and $\mathbb{F}_2$, respectively. Those formulas are given in Table 11. Specifically, in the latter situation we do not need to compute the coefficients corresponding to α in the entries of W. When we avoid such computation we indicate so by putting the operation within square brackets, i.e., $[\,]_{const}$.

Complexity. We obtain the recursive form of the complexities by adding the cost shown in Tables 8, 9, 10 and 11. We then derive two kinds of non-recursive expressions of the complexity. The first one, $\mathcal{C}$, uses

TABLE 9

Multi-evaluation of V

Underlying computations	Entries in $\mathbb{F}_4$	Entries in $\mathbb{F}_2$
	$\#\oplus$	$\#\oplus$
$R'_1 = A_1 + A_2$,	$2n/3$	$n/3$
$R'_2 = \alpha R'_1$ $(= \alpha(A_1 + A_2))$	$n/3$	0
$R'_3 = A_0 + R'_2$ $(= A_0 + \alpha(A_1 + A_2))$	$2n/3$	0
$R'_4 = A_2 + R'_3$ $(= A_2 + A_0 + \alpha(A_1 + A_2))$	$2n/3$	$n/3$
$R'_5 = A_1 + R'_3$ $(= A_1 + A_0 + \alpha(A_1 + A_2))$	$2n/3$	$n/3$
$R'_6 = A_0 + R'_1$ $(= A_0 + A_1 + A_2)$	$2n/3$	$n/3$
Total	$11n/3$	$4n/3$

TABLE 10

Products

Underlying computations	Entries in $\mathbb{F}_4$		Entries in $\mathbb{F}_2$	
	$\#\oplus$	$\otimes$	$\#\oplus$	$\otimes$
$Q_0 = R_1 \cdot A_0$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
$Q_1 = R_7 \cdot R'_6$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
$Q_2 = R_5 \cdot R'_4$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$
$Q_3 = R_6 \cdot R'_5$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$
$Q_4 = R_8 \cdot A_2$	$\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)$	$\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$
Total	$5\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$5\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$	$3\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3) + 2\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$	$3\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3) + 2\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$

a multiplication in $\mathbb{F}_4$ with a cost of 3 bit additions and 4 bit multiplications. The second one, denoted as $\mathcal{C}'$, uses a multiplication in $\mathbb{F}_4$ with a cost of 4 bit additions and 3 bit multiplications. Each expression is obtained by directly applying the formula of Lemma 1 or Lemma 2. The resulting complexities are shown in Table 12.

Delay. Now, we evaluate the delay of the two multipliers. To this end, we determine the delay of each part of the computation:

- The delay of the multi-evaluations of A and T is equal to $4D_\oplus$ when the entries are in $\mathbb{F}_4$ and the delay is $2D_\oplus$ when the entries are in $\mathbb{F}_4$.
- The delay of the recursive multiplications is, in both $\mathbb{F}_4$ and $\mathbb{F}_2$, equal to $\mathcal{D}_{\mathbb{F}_4}(n/3)$.
- The delay of the reconstruction is $4D_\oplus$ in the case of $\mathbb{F}_4$ and $3D_\oplus$ in the case of $\mathbb{F}_2$.

We then obtain the following delays for the TMVP in $\mathbb{F}_4$ and $\mathbb{F}_2$

$$\mathcal{D}_{\mathbb{F}_4}(n) = (8\log_3(n) + 2)D_\oplus + D_\otimes \text{ and } \mathcal{D}_{\mathbb{F}_2}(n) = (8\log_3(n) - 1)D_\oplus + D_\otimes.$$

TABLE 11

Reconstruction

Reconstruction with entries in $\mathbb{F}_4$	
Underlying computations	$\#\oplus$
$U_1 = Q_2 + Q_3$,	$2n/3$
$U_2 = \alpha U_1$ $(= \alpha(Q_2 + Q_3))$	$n/3$
$W_2 = Q_0 + Q_1 + U_1$ $(= Q_0 + Q_1 + Q_2 + Q_3)$	$4n/3$
$U_3 = Q_1 + U_2$ $(= Q_1 + \alpha(Q_2 + Q_3)))$	$2n/3$
$W_1 = Q_3 + U_3$ $(= Q_1 + Q_3 + \alpha(Q_2 + Q_3))$	$2n/3$
$W_0 = Q_2 + Q_4 + U_3$ $(= Q_1 + Q_2 + Q_4 + \alpha(Q_2 + Q_3))$	$4n/3$
Total	$15n/3$

Reconstruction with entries in $\mathbb{F}_2$	
Underlying computations	$\#\oplus$
$U_1 = Q_2 + Q_3$,	$2n/3$
$U_2 = [\alpha U_1]_{const}$	0
$W_2 = [Q_1 + Q_2 + U_1]_{const}$	$2n/3$
$U_3 = [Q_1 + U_2]_{const}$	$n/3$
$W_1 = [Q_3 + U_3]_{const}$	$n/3$
$W_0 = [Q_2 + Q_4 + U_3]_{const}$	$2n/3$
Total	$7n/3$

The symbol $[\,]_{const}$ means that we perform only the operation on bit corresponding to α^0.

TABLE 12

Complexity of 3-way split formulas for TMVP

	Entries in $\mathbb{F}_4$		Entries in $\mathbb{F}_2$	
Recur.	$5\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ $+56n/3-15$	$5\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$	$2\mathcal{S}_{\mathbb{F}_4,\oplus}(n/3)$ $+3\mathcal{S}_{\mathbb{F}_2,\oplus}(n/3)+24n/3-6$	$3\mathcal{S}_{\mathbb{F}_2,\otimes}(n/3)$ $+2\mathcal{S}_{\mathbb{F}_4,\otimes}(n/3)$
Non-recur. $\mathcal{C}$	$\frac{109}{4}n^{\log_3(5)}-28n+\frac{15}{4}$	$4n^{\log_3(5)}$	$\frac{109}{4}n^{\log_3(5)}-\frac{53}{2}n$ $-\frac{32}{3}n\log_3(n)-\frac{3}{4}$	$4n^{\log_3(5)}-3n$
Non-recur. $\mathcal{C}'$	$\frac{113}{4}n^{\log_3(5)}-28n+\frac{15}{4}$	$3n^{\log_3(5)}$	$\frac{113}{4}n^{\log_3(5)}-\frac{55}{2}n$ $-\frac{32}{3}n\log_3(n)-\frac{3}{4}$	$3n^{\log_3(5)}-2n$

6 COMPLEXITY COMPARISON

The resulting complexities of the various approaches presented in this paper are reported in Table 13. Specifically we have reported the complexity of the polynomial multiplication with six recursive multiplications of [11] (reviewed in Table 1), then of Fan *et al.* [6] and our proposed formula (Table 4). We have also reported the complexities of Bernstein's formula with five recursive multiplication (reviewed in Subsection 2.3) and our formula based on field extension (Section 4). We have finally reported the complexities of the 3-way formulas with six recursive multiplications of [5] for TMVP and our proposed formulas with five recursive multiplications for TMVP given in Section 5.

We first compare the formulas with six recursive multiplications. Considering the complexities shown in Table 13, we remark that our proposed 3-way formulas with 6 recursive multiplication improves the space complexity of the previously known formulas. The delay is the same as the delay of the approach based on [11], but higher than the delay of the formula of [6].

Concerning the formulas with five recursive multiplications for polynomial multiplication our approach

TABLE 13

Complexities of different polynomial multiplication approaches considered in this article.

Operation	Algorithm/Formula	$\mathcal{S}_{\oplus}(n)$	$\mathcal{S}_{\otimes}(n)$	Delay
Pol. Mul. with six rec. mult.	Sunar [11]	$5.33n^{\log_3(6)} - 7.33n + 2$	$n^{\log_3(6)}$	$4\log_3(n)D_X$
	Fan [6]	$5.33n^{\log_3(6)} - 7.33n + 2$	$n^{\log_3(6)}$	$3\log_3(n)D_X$
	Proposed formulas of Table 4	$5.27n^{\log_3(6)} - 6.67n + 1.4$	$n^{\log_3(6)}$	$4\log_3(n)D_X$
Pol. Mul. with five rec. mult.	Bernstein [1] ($\mathcal{C}$ in (6))	$18.5n^{\log_3(5)} - 21.5n + 3$	$7n^{\log_3(5)} - 4n - 2$	$(8\log_3(n) + \frac{3n}{2} - \frac{3}{2})D_{\oplus} + D_{\otimes}$
	$\mathcal{C}$ from (13)	$26.75n^{\log_3(5)} - 9.67n\log_3(n)$ $-27.5n + 0.75$	$4n^{\log_3(5)} - 3n$	$10\log_3(n)D_{\oplus} + D_{\otimes}$
	$\mathcal{C}'$ from (13)	$27.75n^{\log_3(5)} - 9.67n\log_3(n)$ $-28.5n + 0.75$	$3n^{\log_3(5)} - 2n$	$10\log_3(n)D_{\oplus} + D_{\otimes}$
TMVP mult.	Fan-Hasan [5]	$4.79n^{\log_3(6)} - 5n + 0.2$	$n^{\log_3(6)}$	$3\log_3(n)D_X$
	Proposed formula for TMVP ($\mathcal{C}$ in Table 12)	$27.25n^{\log_3(5)} - 10.67n\log_3(n)$ $-26.5n - 0.75$	$4n^{\log_3(5)} - 3n$	$(8\log_3(n) + 2)D_{\oplus} + D_{\otimes}$
	Proposed formula for TMVP ($\mathcal{C}'$ in Table 12)	$28.25n^{\log_3(5)} - 10.67n\log_3(n)$ $-27.5n - 0.75$	$3n^{\log_3(5)} - 2n$	$(8\log_3(n) - 1)D_{\oplus} + D_{\otimes}$

has space complexities for the two cases $\mathcal{C}$ and $\mathcal{C}'$ that are higher than the complexities of Bernstein's approach. Specifically, it can be seen from Table 13 that 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 (18.5+7):(26.75+4), i.e., 1:1.2. 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*.

Concerning the formulas for Toeplitz matrix vector product, to the best of our knowledge, this is the first time that an explicit formula with five recursive multiplication is being proposed. The previous best known method is the one in [5] which requires six recursive multiplications, and thus is asymptotically of higher complexity than the proposed approach.

7 Extension to less constrained sizes

In our previous discussion we have assumed that $n = 3^j$. Here we relax the constrained so that its value are closer to those used in practical cryptosystems. To this end, first we consider a specific situation: we would like to multiply two polynomials or a Toeplitz matrix and a vector which are of size $n = 2 \cdot 3^i$ and have their entries in $\mathbb{F}_2$. We start with the case of polynomial multiplication.

Polynomial multiplication of size $n = 2 \cdot 3^i$. We want to perform a single multiplication $A \times B$, where A and

B are of degree $n-1$ and $n = 2 \cdot 3^i$. A direct approach to perform this operation consists of first applying the Karatsuba formula which breaks this multiplication into three multiplications of polynomial of degree $3^i - 1$. Since the size of each of the new polynomial is a power of 3, in these three multiplications we use Bernstein's approach. The cost of the multiplication of A and B is 3 times the complexity of Bernstein's algorithm for polynomial of degree $(n/2-1)$ plus $7n/2-3$ bit additions for the Karatsuba computation. The resulting complexity is given in Table 14.

Here we propose to perform this multiplication in a different fashion. We first split each of 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 two multiplications

$$\begin{aligned} P &= (A_0 + A_1\alpha)(B_0 + B_1\alpha), \text{ which is a product of polynomials in } \mathbb{F}_4[X], \\ P' &= A_1 B_1, \text{ which is a product of polynomials in } \mathbb{F}_2[X]. \end{aligned}$$

If we expand the product which defines P and if we denote $P = P_0 + \alpha P_1$ with $P_0, P_1 \in \mathbb{F}_2[X]$, we obtain that $P_0 = A_0B_0 + A_1B_1$ and $P_1 = A_0B_1 + A_1B_0 + A_1B_1$. We can then remark that

$$\begin{aligned} AB &= (A_0 + X^{n/2}A_1)(B_0 + X^{n/2}B_1) \\ &= P_0 + P' + (P_1 + P')X^{n/2} + P'X^n. \end{aligned}$$

Consequently, we can compute $C = AB$ through one polynomial multiplication of degree $n/2-1$ in $\mathbb{F}_4[X]$ and one multiplication of polynomials of degree $n/2-1$ in $\mathbb{F}_2[X]$ plus $3n-4$ bit additions for the final reconstruction. Using the complexity $\mathcal{C}$ (resp. $\mathcal{C}'$) in (11) for a multiplication in $\mathbb{F}_4[X]$ and the complexity $\mathcal{C}$ (resp. $\mathcal{C}'$) in (13) for a multiplication in $\mathbb{F}_2[X]$ we obtain the complexity $\mathcal{C}$ (resp. $\mathcal{C}'$) given in the upper rows of Table 14.

TABLE 14

Complexities for polynomial multiplication and TMVP of size $n = 2 \times 3^i$

Operation	Method	$\mathcal{S}_{\oplus}(n)$	$\mathcal{S}_{\otimes}(n)$
Pol. Mult.	Karatsuba and Bernstein [1] ($\mathcal{C}$ in (6))	$20.1n^{\log_3(5)} - 28.75n + 6$	$7.6n^{\log_3(5)} - 6n - 6$
	Proposed $\mathcal{C}$	$19.38n^{\log_3(5)} - 4.83n\log_3(n) - 25.2n + 4$	$2.9n^{\log_3(5)} - 1.5n$
	Proposed $\mathcal{C}'$	$20.1n^{\log_3(5)} - 4.83n\log_3(n) - 25.7n + 4$	$2.17n^{\log_3(5)} - n$
TMVP	Proposed $\mathcal{C}$	$19.75n^{\log_3(5)} - 27n + 7.5$	$2.90n^{\log_3(5)}$
	Proposed $\mathcal{C}'$	$20.47n^{\log_3(5)} - 27n + 7.5$	$2.17n^{\log_3(5)}$

Toeplitz matrix vector product when $n = 2 \cdot 3^i$.

We consider here the problem to compute the product $T \cdot A$ where T is an $n \times n$ Toeplitz matrix and A is a vector of size $n = 2 \cdot 3^i$. Since n is a divisible by 2, we split the matrix and the vector, and then expand the block expression of the product as follows

$$\begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \cdot \begin{bmatrix} A_0 \\ A_1 \end{bmatrix} = \begin{bmatrix} T_1A_0 + T_0A_1 \\ T_2A_0 + T_1A_1 \end{bmatrix} \tag{21}$$

The matrices T_0, T_1 and T_2 are $n/2 \times n/2$ Toeplitz matrices and A_1, A_2 are vectors of size $n/2$. We then compute $P = (T_1 + \alpha T_2) \cdot A_0$ and $P' = (T_0 + \alpha T_1) \cdot A_1$ using the method given in Section 5 to compute TMVP with entries in $\mathbb{F}_4$. If we decompose $P = P_0 + \alpha P_1$ and $P' = P'_0 + \alpha P'_1$ we remark that

$$P_0 = T_1 \cdot A_0, \;\; P_1 = T_2 \cdot A_0, \;\; P'_0 = T_0 \cdot A_1 \text{ and } \;\; P'_1 = T_1 \cdot A_1.$$

We can then reconstruct

$$T \cdot A = \begin{bmatrix} P_0 + P'_0 \\ P_1 + P'_1 \end{bmatrix}.$$

The complexity to perform the product $T \cdot A$ is then equal to twice the complexity of $\mathcal{C}$ or $\mathcal{C}'$ in Table 12 plus the n bit additions for the reconstruction. The resulting complexities are given in the lower rows of Table 14.

Complexity of multiplication of polynomials of cryptographic sizes.

In Table 15 we give the complexity of polynomial multiplication for $n = 2^i \cdot 3^j$ that have values in the range $[160, 600]$. The complexities correspond to the combination of Karatsuba (cf [1]) and one of the 3-way formulas. To get the complexity based on the proposed formulas where $i \geq 1$ and $j \geq 1$, we apply Karatsuba recursively up to polynomials of size $2 \cdot 3^j$ and then we apply the strategy presented above to multiply such polynomials.

The resulting complexities show that for small fields the formula with six recursive multiplications gives better results in both time and space. The formula with five recursive multiplication becomes competitive when the size is above 500.

Moreover, considering only formulas with five recursive multiplications' the resulting complexity shows that, for $i \geq 1$ and $j \geq 1$ in $n = 2^i \cdot 3^j$ our approach yields better space and time complexities for the considered fields. The fact that for $n = 3^i$ the proposed 3-way formulas with five recursive multiplications have better space complexity than Bernstein's formula, is due to the terms $-9.67n\log_3(n))$ in $\mathcal{C}$ and $\mathcal{C}'$ in Table 15 which are non negligible for the above-mentioned sizes of polynomials.

8 Conclusion

In this paper, we have considered various recursive 3-way split formulas for polynomial multiplication and Toeplitz matrix vector product. Specifically, we have carefully evaluated the cost of the 3-way split formulas of [11] and [1], and have provided a non-recursive form of their complexity. In Table 13 we have also reported the complexity of [6] which has a lower delay by using a slightly different 3-way splitting.

We have also then presented a new set of 3-way split formula for binary polynomial multiplication with six recursive multiplication and a set of 3-way split formula with five recursive multiplications based on field extension. For each set of formulas we have carefully evaluated their cost and have provided a non-recursive form of their complexity. For the proposed five recursive multiplication formulas, we have

TABLE 15

Complexities for binary polynomial multiplication with $n = 2^i \cdot 3^j \in [160, 600]$

Method	$162 = 2^1 \cdot 3^4$			$192 = 2^6 \cdot 3^1$			$216 = 2^3 \cdot 3^3$			$243 = 2^0 \cdot 3^5$			$256 = 2^8 \cdot 3^0$		
	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay
Karat. and Sunar [11]	3888	19512	19	4374	21614	22	5832	29347	21	7776	39667	20	6561	34295	24
Karat. and Fan [6]	3888	19512	15	4374	21614	21	5832	29347	18	7776	39667	15	6561	34295	24
Karat. and Table 4	3888	19399	19	4374	22314	22	5832	29404	21	7776	39283	20	6561	34230	24
Karat. and Bern. [1]	12147	30036	155	15309	35472	29	20655	50397	72	20901	52591	403	6561	34295	24
Karat. and Proposed $\mathcal{C}$	4757	26217	43	7533	30126	28	8271	42765	39	11771	65167	50	6561	34295	24
Karat. and Proposed $\mathcal{C}'$	3588	27386	43	5832	31827	28	6264	44772	39	8889	68049	50	6561	34295	24

Method	$288 = 2^5 \cdot 3^2$			$324 = 2^2 \cdot 3^4$			$384 = 2^7 \cdot 3^1$			$432 = 2^4 \cdot 3^3$			$486 = 2^1 \cdot 3^5$		
	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay
Karat. and Sunar [11]	8748	44013	23	11664	59667	22	13122	66183	25	17496	89550	24	23328	120699	23
Karat. and Fan [6]	8748	44013	21	11664	59667	18	13122	66183	24	17496	89550	21	23328	120699	18
Karat. and Table 4	8748	44698	23	11664	59326	22	13122	68282	25	17496	89721	24	23328	119546	23
Karat. and Bern. [1]	33291	79026	43	36441	91239	158	45927	107757	32	61965	152700	75	62703	159471	406
Karat. and Proposed $\mathcal{C}$	14013	65661	35	14271	79782	46	22599	91719	31	24813	129804	42	24271	143173	53
Karat. and Proposed $\mathcal{C}'$	10692	68982	35	10764	83289	46	17496	96822	31	18792	135825	42	18264	149180	53

Method	$512 = 2^9 \cdot 3^0$			$576 = 2^6 \cdot 3^2$			$648 = 2^3 \cdot 3^4$			$729 = 2^0 \cdot 3^6$			$768 = 2^8 \cdot 3^1$		
	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay	#AND	#XOR	Delay
Karat. and Sunar [11]	19683	104674	27	26244	134050	26	34992	181265	25	46656	243335	24	39366	201232	28
Karat. and Fan [6]	19683	104674	27	26244	134050	24	34992	181265	21	46656	243335	18	39366	201232	27
Karat. and Table 4	19683	104478	27	26244	136106	26	34992	180243	25	46656	240550	24	39366	207530	28
Karat. and Bern. [1]	19683	104674	27	99873	239091	46	109323	275982	161	106457	273392	1140	137781	325956	35
Karat. and Proposed $\mathcal{C}$	19683	104674	27	42039	198996	38	42813	241611	49	60313	355640	60	67797	277842	34
Karat. and Proposed $\mathcal{C}'$	19683	104674	27	32076	208959	38	32292	252132	49	45417	370536	60	52488	293151	34

computed two non-recursive forms of the complexity: one minimizes the number of bit additions and the other one minimizes the number of bit multiplications.

In this paper, for the first time ever we have also reported five recursive product based TMVP formula. Asymptotically this new formula leads to more efficient implementation of TMVP. In addition, we have shown an efficient way to deal with a class of constrained values of n.

9 Aknowledgements

A preliminary version of this work was presented at SAC 2011 [3]. This work was supported in part by an NSERC research grant awarded to Dr. Anwar Hasan.

References

[1] D. J. Bernstein. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336, 2009.

[2] M. Cenk, Ç. Koç, and F. Özbudak. Polynomial Multiplication over Finite Fields Using Field Extensions and Interpolation. In *19th IEEE Symposium on Computer Arithmetic, ARITH 2009*, pages 84–91, 2009.

[3] M. Cenk, C. Negre, and M.A. Hasan. Improved three-way split formulas for binary polynomial multiplication. In *SAC 2011*. SpringerVerlag, 2011. to appear.

[4] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[5] H. Fan and M. A. Hasan. A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields. *IEEE Trans. Computers*, 56(2):224–233, September 2007.

[6] H. Fan, J. Sun, M. Gu, and K.-Y. Lam. Overlap-free Karatsuba-Ofman Polynomial Multiplication Algorithm, May 2007.

[7] A. A. Karatsuba. The Complexity of Computations. In *Proceedings of the Steklov Institute of Mathematics*, volume 211, pages 169–183, 1995.

[8] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[9] D. A. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT*, pages 343–355, 2004.

[10] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology, proceeding's of CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[11] B. Sunar. A generalized method for constructing subquadratic complexity GF(2^k) multipliers. *IEEE Transactions on Computers*, 53:1097–1105, 2004.

[12] A. L. Toom. The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers. *Soviet Mathematics*, 3:714–716, 1963.

[13] S. Winograd. *Arithmetic Complexity of Computations*. Society For Industrial & Applied Mathematics, U.S., 1980.

APPENDIX

In this section we provide a lemma which gives the non-recursive solution to inductive expression used in the complexity evaluation for some formulas.

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*

$$r_n = n\left(e + \frac{fb^\delta}{a - b^\delta} + \frac{d}{a-1}\right) - n^\delta\left(\frac{fb^\delta}{a-b^\delta}\right) + cn\log_b(n) - \frac{d}{a-1}. \tag{22}$$

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

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

$$r_b = ar_1 + fb^\delta + cb + d = ae + fb^\delta + cb + d \tag{23}$$

Now we compare this value of r_b to the value given by (22) and can write

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

Consequently, the formula (22) 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 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.