Five, Six, and Seven-Term Karatsuba-Like Formulae

Peter L. Montgomery

Abstract—The Karatsuba-Ofman algorithm starts with a way to multiply two 2-term (i.e., linear) polynomials using three scalar multiplications. There is also a way to multiply two 3-term (i.e., quadratic) polynomials using six scalar multiplications. These are used within recursive constructions to multiply two higher-degree polynomials in subquadratic time. We present division-free formulae which multiply two 5-term polynomials with 13 scalar multiplications, two 6-term polynomials with 17 scalar multiplications, and two 7-term polynomials with 22 scalar multiplications. These formulae may be mixed with the 2-term and 3-term formulae within recursive constructions, leading to improved bounds for many other degrees. Using only the 6-term formula leads to better asymptotic performance than standard Karatsuba. The new formulae work in any characteristic, but simplify in characteristic 2. We describe their application to elliptic curve arithmetic over binary fields. We include some timing data.

Index Terms—Karatsuba, Karatsuba-Ofman, polynomial multiplication, characteristic 2, binary fields, Galois fields, elliptic curve arithmetic.

---◆---

1 INTRODUCTION

P OLYNOMIAL arithmetic has many applications. Integer arithmetic algorithms are adaptations of polynomial arithmetic algorithms, with the complication that the integer algorithms worry about carries. Finite fields $GF(p^m)$ are important to cryptography; when p is prime and $m > 1$, field elements are represented by polynomials over the base field $GF(p)$. Computer algebra systems manipulate high and low-degree polynomials, often in multiple variables.

Polynomial addition and subtraction algorithms have little interest since output coefficients can be computed individually, in fixed time. The "schoolbook" way to multiply two n-term polynomials (or two n-digit integers) multiplies each coefficient of one input by each coefficient of the other. This takes $O(n^2)$ steps, which is quadratic in the input size; we can do better. Subquadratic polynomial multiplication algorithms lead to subquadratic times for polynomial division, integer multiplication, and integer division [1, chapter 8].

One subquadratic polynomial multiplication algorithm is Karatsuba-Ofman or, simply, Karatsuba [4, section 4.3.3]. Starting with a scheme which multiplies two 2-term (i.e., linear) polynomials with three scalar multiplications, it multiplies two 2^k-term polynomials (i.e., degree at most $2^k - 1$) with 3^k scalar multiplications. Denoting $n = 2^k$, it multiplies two n-term polynomials with $3^k = n^c$ scalar multiplications, where $c = \log_2 3 \sim 1.585$. The counts of scalar additions and subtractions are also $O(3^k)$, so the overall asymptotic cost is $O(n^c)$. Since $c < 2$, this beats the schoolbook algorithm.

Karatsuba can also utilize a scheme for multiplying two 3-term (i.e., quadratic) polynomials using six scalar multiplications.

Weimerskirch and Paar [7] give a detailed account of the classical Karatsuba algorithm and its variations.

Karatsuba works best when the input lengths are a power of 2, perhaps times a small power of 3. This is often false in practice, although we will assume the two input lengths are equal. One workaround pads the two input polynomials with leading zero coefficients. This meets the $O(n^c)$ asymptotic bound, albeit with higher implied constant. It is desirable to have fast, specialized, formulae for small non-power-of-2 lengths. We present such methods for $n = 5, 6, 7$. The new formulae use fewer multiplications than those in [7].

We show how to save a multiplication for many odd values of n.

If the $n = 6$ formula is used recursively, its asymptotic cost becomes $n^{c'}$, where $c' = \log_6 17 \sim 1.581$, beating the $c = \log_2 3$ exponent.

Although the new formulae have integer coefficients as large as 6, these coefficients reduce to 0 or 1 in characteristic 2. We describe an application to arithmetic in $GF(2^m)$ fields. We include some timing data.

2 MINIMUM MULTIPLICATIONS FUNCTION

Let $a_0 + a_1 X$ and $b_0 + b_1 X$ be two linear polynomials over a ring R. The schoolbook algorithm

$$(a_0 + a_1 X)(b_0 + b_1 X)$$
$$= a_0 b_0 + (a_0 b_1 + a_1 b_0) X + a_1 b_1 X^2 \tag{1}$$

demonstrates that we can multiply these polynomials with four scalar multiplications, namely, the ring products $a_i b_j$, where $0 \le i \le 1$ and $0 \le j \le 1$.

We can do this polynomial product with three scalar multiplications by rewriting the X^1 coefficient as

● *The author is with Microsoft Corporation, One Microsoft Way, Redmond, WA 98052. E-mail: petmon@microsoft.com.*

$$a_0b_1 + a_1b_0 = (a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1.$$

Since we use a_0b_0 and a_1b_1 elsewhere, this saves a multiplication (one new multiplication and some new additions, but two multiplications eliminated). We can summarize the revised formula as

$$
\begin{aligned}
&(a_0 + a_1X)(b_0 + b_1X) \\
&= a_0b_0 + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1)X \\
&\quad + a_1b_1X^2 \\
&= a_0b_0(1 - X) \\
&\quad + (a_0 + a_1)(b_0 + b_1)X \\
&\quad + a_1b_1(X^2 - X).
\end{aligned}
\tag{2}
$$

The factor $1 - X = 1X^0 - 1X^1$ after a_0b_0 means, for example, that the product a_0b_0 appears with a coefficient of 1 in the constant X^0 term, a coefficient of -1 in the X^1 term, and nowhere else. Although it is convenient to refer to an element 1 here, multiplication by an integer is meaningful even if R lacks a multiplicative identity.

For $n = 3$, we present a family of formulae for multiplying two quadratics. We can check that

$$
\begin{aligned}
&(a_0 + a_1X + a_2X^2)(b_0 + b_1X + b_2X^2) \\
&= a_0b_0(C + 1 - X - X^2) \\
&\quad + a_1b_1(C - X + X^2 - X^3) \\
&\quad + a_2b_2(C - X^2 - X^3 + X^4) \\
&\quad + (a_0 + a_1)(b_0 + b_1)(-C + X) \\
&\quad + (a_0 + a_2)(b_0 + b_2)(-C + X^2) \\
&\quad + (a_1 + a_2)(b_1 + b_2)(-C + X^3) \\
&\quad + (a_0 + a_1 + a_2)(b_0 + b_1 + b_2)C
\end{aligned}
\tag{3}
$$

for an arbitrary polynomial C with integer coefficients. The right side of (3) has seven ring products, but we can choose C so one of these products is not needed. For example, choosing $C = X^2$ avoids the need to compute $(a_0 + a_2)(b_0 + b_2)$, leaving six scalar multiplications. This beats the nine needed by the schoolbook algorithm.

2.1 $M(n)$ **Definition**

Identities (2) and (3) prompt us to investigate the complexity of polynomial multiplication.

The inputs and outputs will be in $R[X]$, meaning the polynomial coefficients are in a ring R and the indeterminate is X. We assume this indeterminate commutes with the coefficients a_i and b_j when we replace $(a_1X)b_0$ by $(a_1b_0)X$ in (1), but do not assume the ring R is commutative: a_1b_0 need not equal b_0a_1.

Given a positive integer n, let $M(n)$ denote the minimum number of scalar multiplications needed to multiply two n-term polynomials,

$$a(X) = \sum_{i=0}^{n-1} a_iX^i \quad \text{and} \quad b(X) = \sum_{j=0}^{n-1} b_jX^j, \tag{4}$$

in X, over a ring R. To simplify later analysis, we impose two restrictions on the operands to the multiplications:

1. ($\mathbb{Z}$-linearity) The first operand of each ring multiplication is a $\mathbb{Z}$-linear combination of the a_i (meaning a linear combination with integer coefficients), and the second operand is a $\mathbb{Z}$-linear combination of the b_j.
2. One ring multiplication used is the constant term product a_0b_0.

Restriction 1 reflects a pattern we observe in (2) and (3). One consequence is any operand to the ring multiplications can be evaluated by repeated ring additions (by which terminology we include subtractions).

Since a_0b_0 is the constant term of the product, its value must be computed somehow. We want to preclude an optimal formula which uses

$$a_0b_0 = 3(\text{one product}) - 4(\text{another product}),$$

in which both products on the right are reused elsewhere. Here, we could not easily substitute a_0b_0 in place of another needed product without introducing a division by 3 or 4 at other uses of that product.

The $n = 1$ case of $M(n)$ multiplies two scalars (degree 0 polynomials), so the constant term product a_0b_0 is the sole operation. Formula (2) for $n = 2$ uses the a_0b_0 product. Formula (3) lacks an a_0b_0 product if $C = X^2 + X - 1$, but other choices for C retain the a_0b_0 term. These observations show

$$M(1) = 1, \qquad M(2) \le 3, \qquad M(3) \le 6.$$

This paper gives upper bounds on $M(n)$. The constructions will satisfy Restrictions 1 and 2. The bounds remain valid in contexts where these restrictions are lifted.

2.2 $M(n)$ **for Composite** n

The power of Karatsuba-Ofman stems from recursion. Specifically, if m and n are positive integers, then [7, sections 4.1 and 4.3]

$$M(mn) \le M(m)M(n) \quad (m, n \ge 1). \tag{5}$$

We illustrate when $m = 3$ and $n = 2$. Given two polynomials,

$$
\begin{aligned}
a(X) &= a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 \\
b(X) &= b_0 + b_1X + b_2X^2 + b_3X^3 + b_4X^4 + b_5X^5,
\end{aligned}
$$

each with mn coefficients in a ring R, introduce $Y = X^n = X^2$. Regroup the coefficients of a and b:

$$
\begin{aligned}
a(X, Y) &= (a_0 + a_1X) + (a_2 + a_3X)Y \\
&\quad + (a_4 + a_5X)Y^2 \\
b(X, Y) &= (b_0 + b_1X) + (b_2 + b_3X)Y \\
&\quad + (b_4 + b_5X)Y^2.
\end{aligned}
$$

To compute the multivariate polynomial product $a(X, Y)b(X, Y)$, view its inputs as polynomials in Y whose coefficients are linear polynomials in the ring $R[X]$. The product of two quadratics in Y needs at most $M(3)$ multiplications of linear polynomials in the coefficient ring $R[X]$. Each of these can be done with $M(2)$ multiplications in R, so the multivariate product can be done with $M(3)M(2) \le 6 \cdot 3 = 18$ multiplications in R.

This argument needs the $\mathbb{Z}$-linearity assumption 1 of Section 2.1. The coefficients of Y^0 through Y^2 in $a(X, Y)$ and $b(X, Y)$ are linear (i.e., degree ≤ 1) in X, so any $\mathbb{Z}$-linear combination of these coefficients will itself be linear in X rather than an arbitrary element of the ring $R[X]$.

The algorithm for multiplying two quadratic polynomials in Y is assumed to use the constant-term product $(a_0 + a_1 X)(b_0 + b_1 X)$ as one of its ring multiplications. In turn, the algorithm for multiplying these linear polynomials in X will use $a_0 b_0$ as one of its multiplications. This shows that the mn-term algorithm will satisfy the constant term assumption 2 if the m-term and n-term algorithms satisfy that assumption.

Likewise, the $\mathbb{Z}$-linearity assumption is satisfied by the mn-term algorithm if the m-term and n-term algorithms satisfy that assumption. For example, the first operand to each $R[X]$ product is a $\mathbb{Z}$-linear combination of the polynomials $a_0 + a_1 X$, $a_2 + a_3 X$, and $a_4 + a_5 X$. The first operand to each multiplication in R is a $\mathbb{Z}$-linear combination of the X^0 and X^1 coefficients of one of these linear polynomials.

Later, reduce the multivariate product by substituting $Y = X^2$ and combining like terms. This step requires only ring additions and does not alter the constant term.

Two corollaries of (5) are $M(4) \leq 9$ and $M(9) \leq 36$.

2.3 $M(n)$ for Odd n

If $n = 2m + 1$, where $m \geq 1$, two input polynomials $a(X)$ and $b(X)$ of degree at most $n - 1$ can be written as

$$a(X) = a_0(X) + X^m a_1(X);$$
$$b(X) = b_0(X) + X^m b_1(X),$$

where $a_0(X)$ and $b_0(X)$ have degree at most $m - 1$, while $a_1(X)$ and $b_1(X)$ have degree at most m. We can get $a(X)b(X)$ from the three products

$$a_0(X)b_0(X) \quad \text{and} \quad a_1(X)b_1(X)$$
$$\text{and} \quad (a_0(X) + a_1(X))(b_0(X) + b_1(X)).$$

The first product has operands of degree at most $m - 1$, whereas the other two products have operands of degree at most m. This shows [7, Section 4.1]

$$M(2m + 1) \leq M(m) + 2M(m + 1).$$

We can do better by forming the three products

$$a_0(X)b_0(X) \quad \text{and} \quad a_1(X)b_1(X)$$
$$\text{and} \quad (Xa_0(X) + a_1(X))(Xb_0(X) + b_1(X)).$$

The degrees of the three needed polynomial products are unchanged, but the last two products share the constant term $a_1(0)b_1(0)$. That computation is being done twice and one multiplication is redundant. Therefore,

$$M(2m + 1) \leq M(m) + 2M(m + 1) - 1 \quad (m \geq 1). \quad (6)$$

In particular, (6) shows $M(5) \leq 3 + 2 \cdot 6 - 1 = 14$ and $M(7) \leq 6 + 2 \cdot 9 - 1 = 23$. These bounds beat the ones in Appendix A of [7]. We will soon improve these to $M(5) \leq 13$ and $M(7) \leq 22$.

3 NEW FORMULAE FOR $n = 5, 6, 7$

This work began while implementing elliptic curves over binary fields—see Section 5. Binary fields have characteristic 2, meaning $1 + 1 = 0$. We desired a short algorithm for multiplying degree-4 binary polynomials. Later, we extended the work to degrees 5 and 6.

3.1 Discovery Process for Characteristic 2

Suppose we want to multiply two n-term polynomials (4) in X with indeterminate coefficients a_i, b_j over the base field GF(2). All $2n - 1$ coefficients of the product polynomial $a(X)b(X) \bmod 2$ are elements of a vector space V_n over GF(2) of dimension $n(n + 1)/2$. That space is generated by the formal products $a_i b_i$, where $0 \leq i < n$, and by $a_i b_j + a_j b_i$, where $0 \leq i < j < n$.

Obvious elements of V_n include products such as $(a_1 + 2a_3 - a_4)(b_1 + 2b_3 - b_4)$, wherein the subscripts and coefficients on the as match those on the bs. Working over GF(2), where coefficients must be 0 or 1, there are $2^n - 1$ such products of nonempty sums. We desire to select as few such products as possible and still span the subspace generated by the coefficients of $a(X)b(X)$.

When $n = 5$, there are $\binom{31}{13} \sim 2.1 \cdot 10^8$ ways to select 13 of these products. Given a selection, we check whether its 13 products generate a subspace which has all $2n - 1 = 9$ needed outputs.

For $n = 6$ and $n = 7$, a search of all $\binom{63}{17} \sim 1.0 \cdot 10^{15}$ or $\binom{127}{22} \sim 2.5 \cdot 10^{24}$ possibilities would take too long. Knowing that $a_0 b_0$ and $a_{n-1}b_{n-1}$ and $(a_0 + a_1 + \cdots + a_{n-1})(b_0 + b_1 + \cdots + b_{n-1})$ must be in the span of the chosen products (in order to get proper behavior at $X = 0$, $X = \infty$, and $X = 1$, respectively), we included these three products automatically. This reduces the $n = 6$ search space size to $\binom{60}{14} \sim 1.7 \cdot 10^{13}$ possibilities, a big reduction but still large.

Next, we imposed symmetries on the output. We required a formula to appear unchanged when replacing X by $1/X$ (and multiplying both sides by X^{2n-2} to clear denominators). This means, for example, using a product $(a_{n-1-i} + a_{n-1-j})(b_{n-1-i} + b_{n-1-j})$ whenever we use $(a_i + a_j)(b_i + b_j)$. For $n = 6$, four of the 60 remaining potential products are invariant under this symmetry and 56 are not. Since the nonsymmetric products are being required to occur in pairs, the search space has been reduced to $4 + 56/2 = 32$ products (or pairs of products) rather than 60 products, and the search becomes feasible. The $n = 7$ search proceeded similarly, but took much longer.

The searches yielded multiple spanning sets valid in characteristic 2. For $n = 6$ and $n = 7$, some pairs of spanning sets differed by only one or two elements. We observed this phenomenon once before, when (3) could use any six of the seven potential products. As in (3), we introduce a polynomial C which may be freely chosen and through which one may force a particular coefficient to zero.

3.2 Extension to Other Characteristics

Primarily for aesthetic reasons, the formulae were later modified to work in other characteristics. The adaptation used the Maple symbolic calculator. No attempt was made to optimize the formulae for these other characteristics,

such as trying to minimize the sum of the absolute values of the coefficients.

In the $n = 5$ case, suppose we know the mod 2 version of (7) (see the next section) and want to generalize it. We know that the 13 formal products $(a_0 + \cdots + a_4)(b_0 + \cdots + b_4)$ through $a_0 b_0$ span a space including $(a_0 + \cdots + a_4 X^4)(b_0 + \cdots + b_4 X^4)$ over $\mathrm{GF}(2)[X]$ (polynomials in X with coefficients modulo 2) and want to adjust these 13 generators (perhaps by changing signs of some summands) so this works over $\mathbb{Z}[X]$.

The last four generators in (7) are $a_0 b_0$, $a_1 b_1$, $a_3 b_3$, and $a_4 b_4$. These look so simple that we try to leave them unchanged. Another modulo-2 product is $(a_0 + a_1)(b_0 + b_1)$. Whether we leave this alone or change it to $(a_0 - a_1)(b_0 - b_1)$, we will generate the same space (given the decision to include $a_0 b_0$ and $a_1 b_1$). Likewise, it won't matter whether we flip the signs in $(a_0 + a_4)(b_0 + b_4)$ and $(a_3 + a_4)(b_3 + b_4)$.

The product $(a_0 + \cdots + a_4)(b_0 + \cdots + b_4)$ was left unchanged (with only $+$ signs) since it must be in the span when $X = 1$. It remained to choose the signs within the five remaining generators:

$$(a_0 \pm a_2 \pm a_3 \pm a_4)(b_0 \pm b_2 \pm b_3 \pm b_4),$$
$$(a_0 \pm a_1 \pm a_2 \pm a_4)(b_0 \pm b_1 \pm b_2 \pm b_4),$$
$$(a_0 \pm a_1 \pm a_3 \pm a_4)(b_0 \pm b_1 \pm b_3 \pm b_4),$$
$$(a_0 \pm a_2 \pm a_3)(b_0 \pm b_2 \pm b_3),$$
$$(a_1 \pm a_2 \pm a_4)(b_1 \pm b_2 \pm b_4).$$

Using Maple, we gave symbolic names to these signs, while maintaining symmetries if we interchange as and bs or if we replace each a_i by a_{n-i} and each b_i by b_{n-i}. For example, the last two generators became

$$(a_0 + s_1 a_2 + s_2 a_3)(b_0 + s_1 b_2 + s_2 b_3),$$
$$(s_2 a_1 + s_1 a_2 + a_4)(s_2 b_1 + s_1 b_2 + b_4),$$

with s_1, $s_2 \in \{\pm 1\}$ to be determined. Overall, there were seven independent unknown signs.

The degree-8 product polynomial involving the as and bs is assumed to be a linear combination of these 13 generators, with coefficients in $\mathbb{Z}[X]$ (namely, the parenthesized polynomials in X on the right of (7)). We gave names to the 13 polynomial coefficients. Equating coefficients of each $a_i b_j$, where $0 \le i \le j \le 4$, gave 15 linear equations in the 13 unknown polynomials. We found a choice of signs which allowed everything to be satisfied. Happily, all 13 polynomials had integer (not rational) coefficients.

3.3 Product of Quartic Polynomials

Formula (7) illustrates how to multiply two five-term (degree-4) polynomials with 13 base ring multiplications.

$$\begin{aligned}
&(a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4) \\
&(b_0 + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4) \\
&= (a_0 + a_1 + a_2 + a_3 + a_4) \\
&\quad (b_0 + b_1 + b_2 + b_3 + b_4)(X^5 - X^4 + X^3) \\
&+ (a_0 - a_2 - a_3 - a_4)(b_0 - b_2 - b_3 - b_4) \\
&\quad (X^6 - 2X^5 + 2X^4 - X^3) \\
&+ (a_0 + a_1 + a_2 - a_4)(b_0 + b_1 + b_2 - b_4) \\
&\quad (-X^5 + 2X^4 - 2X^3 + X^2) \\
&+ (a_0 + a_1 - a_3 - a_4) \\
&\quad (b_0 + b_1 - b_3 - b_4)(X^5 - 2X^4 + X^3) \\
&+ (a_0 - a_2 - a_3)(b_0 - b_2 - b_3) \\
&\quad (-X^6 + 2X^5 - X^4) \\
&+ (a_1 + a_2 - a_4)(b_1 + b_2 - b_4) \\
&\quad (-X^4 + 2X^3 - X^2) \\
&+ (a_3 + a_4)(b_3 + b_4)(X^7 - X^6 + X^4 - X^3) \\
&+ (a_0 + a_1)(b_0 + b_1)(-X^5 + X^4 - X^2 + X) \\
&+ (a_0 - a_4)(b_0 - b_4) \\
&\quad (-X^6 + 3X^5 - 4X^4 + 3X^3 - X^2) \\
&+ a_4 b_4 (X^8 - X^7 + X^6 \\
&\quad - 2X^5 + 3X^4 - 3X^3 + X^2) \\
&+ a_3 b_3 (-X^7 + 2X^6 - 2X^5 + X^4) \\
&+ a_1 b_1 (X^4 - 2X^3 + 2X^2 - X) \\
&+ a_0 b_0 (X^6 - 3X^5 + 3X^4 - 2X^3 + X^2 - X + 1).
\end{aligned} \tag{7}$$

The operands of the 13 ring multiplications in (7) can be evaluated with 22 ring additions or subtractions by taking advantage of common subexpressions. Starting with a_0 to a_4, one can evaluate

$$\begin{aligned}
&a_0 + a_1, \\
&a_0 - a_4, \\
&a_3 + a_4, \\
&(a_0 + a_1) - (a_3 + a_4), \\
&(a_0 + a_1) + a_2, \\
&a_2 + (a_3 + a_4), \\
&(a_0 + a_1) + (a_2 + a_3 + a_4), \\
&a_0 - (a_2 + a_3 + a_4), \\
&(a_0 + a_1 + a_2) - a_4, \\
&(a_0 - a_2 - a_3 - a_4) + a_4, \\
&(a_0 + a_1 + a_2 - a_4) - a_0
\end{aligned}$$

in sequence and likewise with the bs.

Other repeated subexpressions arise while processing the outputs of the 13 products, although it is hard to count these. For example, the right of (7) includes the terms

$$\begin{aligned}
&a_1 b_1 (-2X^3 - X) \\
&+ a_0 b_0 (-3X^5 + 3X^4 - 2X^3 - X) \\
&= (a_1 b_1 + a_0 b_0)(-2X^3 - X) + 3 a_0 b_0 (X^4 - X^5).
\end{aligned}$$

After investing three ring additions to evaluate $a_1 b_1 + a_0 b_0$ and $3 a_0 b_0$, another $2 + 1 + 1 + 1 = 5$ ring additions suffice to

adjust the coefficients of the output polynomial. We would need $2 + 1 + 3 + 3 + 2 + 1 = 12$ ring additions if we adjusted the output coefficients directly.

These operation counts may be lower in characteristics 2 and 3, wherein all coefficients simplify to 0 or ± 1.

Formula (7) shows $M(5) \leq 13$. This can be used in recursive constructions. For example, (6) with $m = 4$ shows $M(9) \leq M(4) + 2M(5) - 1 \leq 9 + 2 \cdot 13 - 1 = 34$, beating the $M(9) \leq M(3)^2 \leq 6^2 = 36$ bound from (5).

3.4   Product of Quintic Polynomials

Formula (8) illustrates how to multiply two six-term (degree-5) polynomials with 17 base ring multiplications. The parameter C should be an integer polynomial chosen so one product disappears (other than the $a_0 b_0$ product).

$$(a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4 + a_5 X^5)$$
$$(b_0 + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5)$$
$$= (a_0 + a_1 + a_2 + a_3 + a_4 + a_5)$$
$$(b_0 + b_1 + b_2 + b_3 + b_4 + b_5)C$$
$$+ (a_1 + a_2 + a_4 + a_5)(b_1 + b_2 + b_4 + b_5)$$
$$(-C + X^6)$$
$$+ (a_0 + a_1 + a_3 + a_4)(b_0 + b_1 + b_3 + b_4)$$
$$(-C + X^4)$$
$$+ (a_0 - a_2 - a_3 + a_5)(b_0 - b_2 - b_3 + b_5)$$
$$(C - X^7 + X^6 - X^5 + X^4 - X^3)$$
$$+ (a_0 - a_2 - a_5)(b_0 - b_2 - b_5)$$
$$(C - X^5 + X^4 - X^3)$$
$$+ (a_0 + a_3 - a_5)(b_0 + b_3 - b_5)$$
$$(C - X^7 + X^6 - X^5)$$
$$+ (a_0 + a_1 + a_2)(b_0 + b_1 + b_2)$$
$$(C - X^7 + X^6 - 2X^5 + 2X^4 - 2X^3 + X^2)$$
$$+ (a_3 + a_4 + a_5)(b_3 + b_4 + b_5)$$
$$(C + X^8 - 2X^7 + 2X^6 - 2X^5 + X^4 - X^3)$$
$$+ (a_2 + a_3)(b_2 + b_3)$$
$$(-2C + X^7 - X^6 + 2X^5 - X^4 + X^3)$$
$$+ (a_1 - a_4)(b_1 - b_4)(-C + X^4 - X^5 + X^6)$$
$$+ (a_1 + a_2)(b_1 + b_2)$$
$$(-C + X^7 - 2X^6 + 2X^5 - 2X^4 + 3X^3 - X^2)$$
$$+ (a_3 + a_4)(b_3 + b_4)$$
$$(-C - X^8 + 3X^7 - 2X^6 + 2X^5 - 2X^4 + X^3)$$
$$+ (a_0 + a_1)(b_0 + b_1)(-C + X^7$$
$$- X^6 + 2X^5 - 3X^4 + 2X^3 - X^2 + X)$$
$$+ (a_4 + a_5)(b_4 + b_5)(-C + X^9$$
$$- X^8 + 2X^7 - 3X^6 + 2X^5 - X^4 + X^3)$$
$$+ a_0 b_0 (-3C + 2X^7 - 2X^6$$
$$+ 3X^5 - 2X^4 + 2X^3 - X + 1)$$
$$+ a_1 b_1 (3C - X^7 - X^5 + X^4 - 3X^3 + 2X^2 - X)$$
$$\text{(8)}$$

$$+ a_4 b_4 (3C - X^9 + 2X^8 - 3X^7 + X^6 - X^5 - X^3)$$
$$+ a_5 b_5 (-3C + X^{10} - X^9$$
$$+ 2X^7 - 2X^6 + 3X^5 - 2X^4 + 2X^3).$$

This shows $M(6) \leq 17$, beating the $M(6) \leq M(2)M(3) \leq 3 \cdot 6 = 18$ bound from (5).

As with (7), we should consider common subexpressions while counting the ring additions and subtractions. These counts depend on C and R.

Formula (8) can be used recursively to multiply two 6^k-term polynomials with 17^k scalar multiplications. For large n, it multiplies two n-term polynomials in time $O(n^c)$, where $c = \log_6 17 \sim 1.581$. This beats the exponent $\log_2 3 \sim 1.585$ using (2) recursively. However, the difference won't be noticeable until Fast Fourier Transform methods [1, chapter 7], [2], [3] are superior.

3.5   Product of Sextic Polynomials

Formula (9) illustrates how to multiply two seven-term (degree-6) polynomials with 22 base ring multiplications. The parameter C should be an integer polynomial chosen so one product disappears.

$$(a_0 + a_1 X + a_2 X^2 + a_3 X^3 + a_4 X^4$$
$$+ a_5 X^5 + a_6 X^6)$$
$$(b_0 + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4$$
$$+ b_5 X^5 + b_6 X^6)$$
$$= (a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6)$$
$$(b_0 + b_1 + b_2 + b_3 + b_4 + b_5 + b_6)$$
$$(X^7 - X^6 + X^5)$$
$$+ (a_1 + a_2 + a_3 - a_5 - a_6)$$
$$(b_1 + b_2 + b_3 - b_5 - b_6)$$
$$(-C - X^9 + 2X^7 - 3X^6 + 2X^5)$$
$$+ (a_0 + a_1 - a_3 - a_4 - a_5)$$
$$(b_0 + b_1 - b_3 - b_4 - b_5)$$
$$(-C + 2X^7 - 3X^6 + 2X^5 - X^3)$$
$$+ (a_0 - a_2 - a_3 - a_4 + a_6)$$
$$(b_0 - b_2 - b_3 - b_4 + b_6)$$
$$(C + X^9 - 4X^7 + 6X^6 - 4X^5 + X^3)$$
$$+ (a_0 - a_2 - a_3 + a_5 + a_6)$$
$$(b_0 - b_2 - b_3 + b_5 + b_6)$$
$$(X^7 - 2X^6 + 2X^5 - X^3)$$
$$+ (a_0 + a_1 - a_3 - a_4 + a_6)$$
$$(b_0 + b_1 - b_3 - b_4 + b_6)$$
$$(-X^9 + 2X^7 - 2X^6 + X^5)$$
$$+ (a_1 + a_2 - a_4 - a_5)(b_1 + b_2 - b_4 - b_5)$$
$$(C + X^9 - 3X^7 + 4X^6 - 3X^5 + X^3)$$
$$+ (a_0 + a_1 - a_5 - a_6)(b_0 + b_1 - b_5 - b_6)C$$
$$+ (a_0 + a_1)(b_0 + b_1)$$
$$(X^9 - 5X^7 + 6X^6 - 4X^5 + X^3 + X)$$
$$\text{(9)}$$

$+ (a_0 - a_2)(b_0 - b_2)$

$\quad (-C - X^9 + 4X^7 - 5X^6 + 3X^5 - X^2)$

$+ (a_0 - a_4)(b_0 - b_4)(X^7 - 2X^6 + 2X^5 - X^4)$

$+ (a_1 + a_3)(b_1 + b_3)(X^7 - X^6 + X^4 - X^3)$

$+ (a_2 - a_6)(b_2 - b_6)(-X^8 + 2X^7 - 2X^6 + X^5)$

$+ (a_3 + a_5)(b_3 + b_5)(-X^9 + X^8 - X^6 + X^5)$

$+ (a_4 - a_6)(b_4 - b_6)$

$\quad (-C - X^{10} + 3X^7 - 5X^6 + 4X^5 - X^3)$

$+ (a_5 + a_6)(b_5 + b_6)$

$\quad (X^{11} + X^9 - 4X^7 + 6X^6 - 5X^5 + X^3)$

$+ a_0 b_0 (-2X^7 + 3X^6 - 3X^5$

$\quad + X^4 + X^2 - X + 1)$

$+ a_1 b_1 (X^5 - X^4 + X^2 - X)$

$+ a_2 b_2 (X^8 - 3X^7 + 3X^6 - 2X^5$

$\quad + X^4 - X^3 + X^2)$

$+ a_3 b_3 (C + 2X^9 - X^8 - 5X^7$

$\quad + 8X^6 - 5X^5 - X^4 + 2X^3)$

$+ a_4 b_4 (X^{10} - X^9 + X^8 - 2X^7$

$\quad + 3X^6 - 3X^5 + X^4)$

$+ a_5 b_5 (-X^{11} + X^{10} - X^8 + X^7)$

$+ a_6 b_6 (X^{12} - X^{11} + X^{10} + X^8$

$\quad - 3X^7 + 3X^6 - 2X^5).$

This shows $M(7) \leq 22$. Like (7) and (8), this can be used within recursive constructions.

4 REVISED $M(n)$ BOUNDS FOR SMALL n

Table 1 gives the known bounds on $M(n)$ for some small n, from Appendix A of [7] (smaller of their MUL# count for general recursive KA and for simple recursive KA). The bounds in column 2 of Table 1 can be achieved using (2), (3), (5), and (6) repeatedly (but without the -1 term in (6)). The third column improves those bounds using the improved (6) as well as (7), (8), and (9). The "Why?" column shows a multiplication if the new bound uses (5) and an addition if it uses (6).

The $n^{\log_2 3}$ column gives an idealized multiplication count if that cost formula extended to non-powers-of-2 for n. We observe that columns 3 and 5 remain close for $n \leq 16$, suggesting it will be hard to improve these much more.

5 APPLICATION TO ELLIPTIC CURVES OVER $\mathrm{GF}(2^m)$

For elliptic curve cryptography, NIST [5] recommends certain elliptic curves. Ten of its recommendations are defined over the five binary fields $\mathrm{GF}(2^m)$ for $m = 163, 233, 283, 409, 571$.

Using polynomial bases, an element of $\mathrm{GF}(2^m)$ is represented by a polynomial of degree at most $m - 1$ in a variable α. The polynomial coefficients are in the base field $\mathrm{GF}(2)$ (hence, 0 or 1). In characteristic 2, the square of a field element $a = \sum_{i=0}^{m-1} a_i \alpha^i$ is

TABLE 1
Upper Bounds for $M(n)$

n	$M(n)$ bound from [7]	New bound	Why?	$n^{\log_2 3}$
1	1	1		1.0
2	3	3	(2)	3.0
3	6	6	(3)	5.7
4	9	9	$2 \cdot 2$	9.0
5	15	13	(7)	12.8
6	18	17	(8)	17.1
7	24	22	(9)	21.8
8	27	27	$2 \cdot 4$	27.0
9	36	34	$4 + 5$	32.5
10	45	39	$2 \cdot 5$	38.4
11	51	46	$5 + 6$	44.7
12	56	51	$2 \cdot 6$	51.3
13	66	60	$6 + 7$	58.3
14	72	66	$2 \cdot 7$	65.5
15	78	75	$7 + 8$	73.1
16	81	81	$2 \cdot 8$	81.0
17	105	94	$8 + 9$	89.2
18	108	102	$3 \cdot 6$	97.6

$$a^2 = \sum_{i=0}^{m-1} \left(a_i \alpha^i \right)^2 = \sum_{i=0}^{m-1} a_i \alpha^{2i}. \tag{10}$$

The right side of (10) has degree at most $2m - 2$ and must be reduced modulo the minimal polynomial for α. This minimal (irreducible) polynomial has degree exactly m. This reduction is fast if the minimal polynomial is sparse—all binary fields in [5] are defined modulo a trinomial or pentanomial.

When doing a general multiplication ab over $\mathrm{GF}(2^m)$, where

$$a = \sum_{i=0}^{m-1} a_i \alpha^i \quad \text{and} \quad b = \sum_{i=0}^{m-1} b_i \alpha^i,$$

with all $a_i, b_i \in \mathrm{GF}(2)$, the two binary polynomials of degree at most $m - 1$ are multiplied, getting a product of degree at most $2m - 2$, which is then reduced modulo the minimal polynomial for α.

Assume one is programming a general purpose 32-bit machine. Pad the two input polynomials a and b with leading zeros so they have $32 \lceil m/32 \rceil$ coefficients (bits) each, storing 32 bits (representing coefficients of consecutive powers of α) per word. Using the methods of Section 2.2, replace the degree $32 \lceil m/32 \rceil - 1$ product by $M(\lceil m/32 \rceil)$ products of 32-coefficient polynomials.

Few RISC architectures have a $\mathrm{GF}(2)$ polynomial multiplication instruction (e.g., given two input polynomials of degree at most 31, output their product of degree at most 62). We must do this step in software. The Karatsuba techniques reduce its frequency of occurrence at the expense of extra 32-bit and 64-bit additions (i.e., exclusive ORs). As noted in Section 3.3, an accurate estimate of this overhead must consider repeated subexpressions. Each exclusive OR is usually very fast.

For the fields in [5], the values of $\lceil m/32 \rceil$ are $6, 8, 9, 13, 18$. Table 1 shows improvements ranging from none for $m = 233$ to 9.1 percent for $m = 409$. On a 64-bit machine, the improvement is 13.3 percent when $m = 283$ and $\lceil m/64 \rceil = 5$.

6   TIMING COMPARISONS

In response to reviewer requests, we include some timing data. Before we describe the experiments, we mention the Toom-Cook method, modified by Zuras.

6.1   Toom-Cook-Zuras Polynomial Multiplication

Toom-Cook [6], [4] multiplies two polynomials P, Q of degree at most $r \geq 0$ by evaluating $P(x_i)$ and $Q(x_i)$ at $2r+1$ distinct points x_i. It takes the pointwise products $P(x_i)Q(x_i)$ and interpolates to find a polynomial R of degree at most $2r$ with $R(x_i) = P(x_i)Q(x_i)$ for all i.

Toom suggests letting the x_i be the integers from 0 to $2r$. Zuras [8] suggests using fractions $x_i = y_i/z_i$ (possibly negative), evaluating the homogeneous polynomial $z_i^r P(y_i/z_i)$.

These schemes won't work over a field with fewer than $2r$ elements, even if we use ∞ for one point of evaluation. Over a commutative ring, no $x_i - x_j$ can be a zero divisor when $i \neq j$.

6.2   Timing Integer Polynomial Multiplication

One experiment multiplied two high-degree integer polynomials, a problem doable by the modified Karatsuba, by Toom-Cook, and by complex Fast Fourier Transforms (FFTs). It was run on one processor of a 1300 MHz Itanium 2. The Itanium was chosen because it has hardware support for 64-bit integers and because a commercial FFT library, specifically SGI's Scientific Computing Software Library (SCSL), was available.

Let k be a power of 2, with $k \leq 2048$. Choose two $5k$-term polynomials $P(X)$, $Q(X)$ with 24-bit integer coefficients (0 to $16,777,215$). The coefficients of the product $P(X)Q(X)$ are bounded by $(2^{24})^2 \cdot 5k \leq 5 \cdot 2^{59}$ and fit into a 64-bit integer.

The modified Karatsuba code lets $Y = X^k$. As in Section 2.2, write $P(X)$ as

$$P(X) = P_0(X)$$
$$+ P_1(X)Y + P_2(X)Y^2 \qquad (11)$$
$$+ P_3(X)Y^3 + P_4(X)Y^4,$$

where $\deg(P_i) \leq k - 1$ for all i. Write $Q(X)$ similarly. The two 5-term polynomials in Y were multiplied using 13 multiplications of polynomials in X, as in (7). These 13 multiplications of k-term polynomials used standard power-of-2 Karatsuba, switching to the straightforward algorithm when the inputs dropped to four terms.

This processing used only additions, subtractions, and multiplications. All arithmetic was modulo 2^{64}. Each coefficient of the final result is known a priori to be in the interval $[0, 2^{64} - 1]$ and must be correct, even if intermediate results overflowed.

The Toom-Cook-Zuras code evaluated a homogeneous variant of (11) at the nine Zuras-suggested rational points

$$Y = 0, \infty, 1, \pm 1/2, \pm 2, 1/3, 3,$$

again with 64-bit integer arithmetic. The resulting pairs of polynomials in X were multiplied using the power-of-2 Karatsuba code.

The Toom-Cook-Zuras code divides by $25200 = 2^4 \cdot 1575$ during interpolation. (Zuras [8, Fig. 5] shows this

TABLE 2
Itanium 2 Timings (μsec) for Integer Polynomial Multiplication

$5k$	Modified Karatsuba (exact)	Toom-Cook-Zuras (mod 2^{60})	SCSL FFT Time	Max error
5	1	1	3	0.08
10	1	1	3	0.12
20	3	2	4	0.25
40	8	7	6	0.62
80	23	19	9	1.81
160	68	50	17	4
320	204	151	30	12
640	597	461	60	15
1280	1790	1383	120	53
2560	5615	3906	268	110
5120	16601	11719	588	219
10240	50781	34180	1670	514
20480	151367	104493	3635	1148

denominator, but has misplaced elements in the first and last columns of the 9×9 inverse.) This division is not a modulo 2^{64} operation: If $25200n_1 \equiv 25200n_2 \pmod{2^{64}}$, we can conclude $n_1 \equiv n_2 \pmod{2^{60}}$ but not modulo 2^{64}. The division used a multiplication by the 64-bit integer $(689 \cdot 2^{64} + 1)/1575$ followed by a 4-bit right shift. That gave the correct least significant 60 bits of each output coefficient. If we had evaluated (11) at the nine integer points $Y = -4, \cdots, 4$, then the interpolation code would divide by $8! = 2^7 \cdot 315$ and the output coefficients would have only 57 correct bits.

To multiply two integer polynomials, the FFT code converted the $5k$ integer coefficients of each input polynomial to double precision and padded them with $5k$ leading zeros. Each length-$10k$ coefficient array was then transformed to double precision complex via SCSL routine **dzfft**. The transform outputs were multiplied pointwise, followed by a reverse transform via SCSL routine **zdfft**.

Table 2 compares the timings for these three methods. Its last column has the maximum observed numerical error in an output coefficient due to the use of floating-point arithmetic. The FFT time omits the overhead for initializing the trigonometric constants.

The modified Karatsuba and Toom-Cook-Zuras methods show little difference for $5k \leq 40$, beyond which the Toom-Cook-Zuras time soon drops to 70 percent of the modified Karatsuba time due to nine products of length k rather than 13 such products.

The FFT method is slowest for small $5k$, but wins for $5k \geq 40$. That is the same point at which the numerical errors in the output coefficients may exceed 0.5, making it hard to accurately round them to integers.

The modified Karatsuba code is never fastest, although it has the virtue of always giving correct 64-bit results.

6.3   Timing $GF(2)$ Polynomial Multiplication

We also timed some polynomial products over the binary field $GF(2)$, like those needed for the elliptic curve computations in Section 5 (but for much higher degrees). For $m = 3, 4, 5, 6, 7$ and n a power of 2, we timed the product of two mn-bit binary polynomials, getting a $2mn$-bit product.

The codes started with one Karatsuba transform of length m, similar to (11), but with degree 4 replaced by

TABLE 3
Pentium 4 Timings (μsec) for *mn*-Bit GF(2) Polynomial Product

n	$m = 3$	$m = 4$	$m = 5$	$m = 6$	$m = 7$
64	3	4	5	7	9
128	7	11	17	20	26
256	22	33	55	61	83
512	64	106	139	183	240
1024	191	289	418	548	741
2048	588	873	1280	1681	2151
4096	1747	2596	3769	4908	6375
8192	5264	7893	11461	14945	19367

$m - 1$. The transforms for $m = 5$, 6, 7 used (7), (8), and (9), with $C = 0$ in the latter two cases. Later, it invoked several polynomial multiplications of length n—those multiplications were done by standard Karatsuba. The program ran on a 1700 MHz Pentium 4, making heavy use of 64-bit MMX data. Microsoft's Visual C++ .NET compiler supports intrinsic functions which let a programmer access the MMX instructions.

Cantor [2] gives an FFT-like algorithm for arithmetic over finite fields and another [3] for arbitrary algebras. His algorithms were not timed.

The times in Table 3 are $(mn/32)^{1.58}c$ microseconds, where c is between 0.13 and 0.14. The times for $5n$, $6n$, and $7n$ fit nicely between those for $4n$ and for $4(2n)$ (from the next line). We observe that the time for $6n$ is slightly better than that for $3(2n)$ because we need 17 rather than 18 products of n-bit polynomials.

7 CONCLUSION

We presented division-free formulae which multiply two polynomials of degree at most 4, 5, or 6, using fewer scalar multiplications than known methods. We gave an improvement valid for many odd lengths. The new formulae may be used in recursive constructions. We presented an application to elliptic curves cryptography. Experiments revealed that the new formulae improved classical Karatsuba on one application, but Karatsuba lost to FFT on another application.

ACKNOWLEDGMENTS

REFERENCES

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
[2] D.G. Cantor, "On Arithmetical Algorithms over Finite Fields," *J. Combinatorial Theory,* Series A 50, pp. 285-300, 1989.
[3] D.G. Cantor and E. Kaltofen, "On Fast Multiplication of Polynomials over Arbitrary Algebras," *Acta Informatica,* vol. 28, pp. 693-701, 1991.
[4] D.E. Knuth, "Seminumerical Algorithms," *The Art of Computer Programming,* vol 2, third ed., Addison-Wesley, 1997.
[5] *Recommended Elliptic Curves for Federal Government Use,* July 1999. Appendix 6 to FIPS publication 186-2, Digital Signature Standard (DSS), US Dept. of Commerce, Nat'l Inst. of Standards and Technology, Jan. 2000, http://www.itl.nist.gov/fipspubs/fip186.htm.
[6] A.L. Toom, "The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers," *Soviet Math.,* vol. 4, pp. 714-716, 1963.
[7] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations," 2003, http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/kaweb.pdf.
[8] D. Zuras, "More on Squaring and Multiplying Large Integers," *IEEE Trans. Computers,* vol. 43, no. 8, pp. 899-908, Aug. 1994.

Peter L. Montgomery was a 1967 Putnam fellow while at the University of California at Berkeley. He worked 17 years at Unisys before receiving the PhD degree from the University of California at Los Angeles in 1992. Now, he's a cryptographer at Microsoft Research in Redmond, Washington. His specialty is computational number theory. He is best known as the inventor of Montgomery multiplication. He has improved integer factorization algorithms.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.