FASTER POLYNOMIAL MULTIPLICATION OVER FINITE FIELDS USING CYCLOTOMIC COEFFICIENT RINGS

DAVID HARVEY AND JORIS VAN DER HOEVEN

Abstract. We prove that for a fixed prime p, polynomials in $\mathbb{F}_p[X]$ of degree n may be multiplied in $O(n \log n \, 4^{\log^* n})$ bit operations. Previously, the best known bound was $O(n \log n \, 8^{\log^* n})$.

1. Introduction

In this paper we present a new complexity bound for multiplying polynomials over finite fields. Our focus is on theoretical bounds rather than practical algorithms. We work in the deterministic multitape Turing model [16], in which time complexity is defined by counting the number of steps, or equivalently, the number of 'bit operations', executed by a Turing machine with a fixed, finite number of tapes. The main results of the paper also hold in the Boolean circuit model.

The following notation is used throughout. For $x \in \mathbb{R}$, we denote by $\log^* x$ the iterated logarithm, that is, the least non-negative integer k such that $\log^{\circ k} x \leqslant 1$, where $\log^{\circ k} x := \log \cdots \log x$ (iterated k times). For a positive integer n, we define $\lg n := \max(1, \lceil \log_2 n \rceil)$; in particular, expressions like $\lg \lg \lg n$ are defined and take positive values for all $n \geqslant 1$. We denote the n-th cyclotomic polynomial by $\phi_n(X) \in \mathbb{Z}[X]$, and the Euler totient function by $\varphi(n)$, i.e., $\varphi(n) := |(\mathbb{Z}/n\mathbb{Z})^*|$.

All absolute constants in this paper are in principle effectively computable. This includes the implied constants in all uses of $O(\cdot)$ notation.

1.1. Statement of main result.

Let $\mathsf{M}(n)$ denote the number of bit operations required to multiply two n-bit integers. We assume that

$$\mathsf{M}(n) = O(n \lg n \, K_{\mathbb{Z}}^{\log^* n}) \tag{1.1}$$

for some constant $K_{\mathbb{Z}} \geqslant 1$. The first bound of this type was proved by Fürer, for some unspecified value of $K_{\mathbb{Z}}$ [8, 9]. Currently the best known value for the constant is $K_{\mathbb{Z}} = 4$ [11].

For a prime p, let $\mathsf{M}_p(n)$ denote the number of bit operations required to multiply two polynomials in $\mathbb{F}_p[X]$ of degree less than n. Our main result is as follows.

Theorem 1.1. *There is a polynomial multiplication algorithm that achieves*

$$\mathsf{M}_p(n) = O(n \lg p \lg(n \lg p) \, 4^{\max(0, \log^* n - \log^* p)} K_{\mathbb{Z}}^{\log^* p}), \tag{1.2}$$

uniformly for all $n \geqslant 1$ and all primes p.

In particular, for fixed p, one can multiply polynomials in $\mathbb{F}_p[X]$ of degree n in $O(n \lg n \, 4^{\log^* n})$ bit operations.

Theorem 1.1 may be generalised in various ways. We briefly mention a few possibilities along the lines of [15, §8] (no proofs will be given). First, we may obtain analogous bit complexity bounds for multiplication in $\mathbb{F}_{p^a}[X]$ and $(\mathbb{Z}/p^a\mathbb{Z})[X]$ for $a \geqslant 1$,

and in $(\mathbb{Z}/m\mathbb{Z})[X]$ for arbitrary $m \geqslant 1$ (see Theorems 8.1–8.3 in [15]). We may also obtain complexity bounds for polynomial multiplication in various algebraic complexity models. For example, we may construct a straight-line program that multiplies two polynomials in $\mathcal{A}[X]$ of degree less than n, for any $\mathbb{F}_p$-algebra $\mathcal{A}$, using $O(n \lg n \, 4^{\log^* n})$ additions and scalar multiplications and $O(n \, 2^{\log^* n})$ nonscalar multiplications (compare with [15, Thm. 8.4]).

Remark 1.2. A previous version of this paper [12] used the algorithm underlying Theorem 1.1 to construct an integer multiplication algorithm achieving $K_{\mathbb{Z}} = 2^{5/2} \approx 5.66$. This result was superseded by a subsequent paper that achieves $K_{\mathbb{Z}} = 4$ using a completely different method [11].

1.2. **Comparison with previous work.** The optimal choice of algorithm for multiplication in $\mathbb{F}_p[x]$ depends very much on the relative size of n and p. If n is not too large compared to p, say $\lg n = O(\lg p)$, then a reasonable choice is *Kronecker substitution*: one lifts the polynomials to $\mathbb{Z}[X]$, packs the coefficients of each polynomial into a large integer (i.e., evaluates at $X = 2^b$ for $b := 2 \lg p + \lg n$), multiplies these large integers, unpacks the resulting coefficients to obtain the product in $\mathbb{Z}[X]$, and finally reduces the output modulo p. This leads to the bound

$$\mathsf{M}_p(n) = O(\mathsf{M}(n \lg p)) = O(n \lg p \lg(n \lg p) K_{\mathbb{Z}}^{\log^*(n \lg p)}). \tag{1.3}$$

Note that (1.2) reduces to this bound in the region $\lg n = O(\lg p)$. To the authors' knowledge, this is the best known asymptotic bound for $\mathsf{M}_p(n)$ in this region.

When n is large compared to p, the situation is starkly different. The Kronecker substitution method leads to poor results, due to coefficient growth in the lifted product: for example, when p is fixed, Kronecker substitution yields

$$\mathsf{M}_p(n) = O(\mathsf{M}(n \lg n)) = O(n (\lg n)^2 K_{\mathbb{Z}}^{\log^* n}).$$

For many years, the best known bound in this regime was that achieved by the algebraic version of the Schönhage–Strassen algorithm [19, 18] (see also [6, Sec. 2.2]), namely

$$\mathsf{M}_p(n) = O(n \lg n \lg \lg n \lg p + n \lg n \, \mathsf{M}(\lg p)). \tag{1.4}$$

The first term arises from performing $O(n \lg n \lg \lg n)$ additions in $\mathbb{F}_p$, and the second term from $O(n \lg n)$ multiplications in $\mathbb{F}_p$. (In fact, this sort of bound holds for polynomial multiplication over quite general rings [7].) For fixed p, this is faster than the Kronecker substitution method by a factor of almost $\lg n$. The main reason for its superiority is that it exploits the modulo p structure throughout the algorithm, whereas the Kronecker substitution method forgets this structure in the very first step.

After the appearance of Fürer's algorithm for integer multiplication, it was natural to ask whether a Fürer-type bound could be proved for $\mathsf{M}_p(n)$, in the case that n is large compared to p. This question was answered in the affirmative by Harvey, van der Hoeven and Lecerf, who gave an algorithm that achieves

$$\mathsf{M}_p(n) = O(n \lg p \lg(n \lg p) \, 8^{\log^*(n \lg p)}),$$

uniformly for all n and p [15]. This is a very elegant bound; however, written in this way, it obscures the fact that the constant 8 plays two quite different roles in the complexity analysis. One source of the value 8 is the constant $K_{\mathbb{Z}} = 8$ arising from the best known *integer* multiplication algorithm at the time the paper was written, but there is also a separate constant $K_{\mathbb{F}} = 8$ arising from the *polynomial*

part of the algorithm. There is no particular reason to expect that $K_{\mathbb{Z}} = K_{\mathbb{F}}$, and it is somewhat of a coincidence that they have the same numerical value in [15].

To clarify the situation, we mention that one may derive a complexity bound for the algorithm of [15] under the assumption that one has available an integer multiplication algorithm achieving (1.1) for some $K_{\mathbb{Z}} \geqslant 1$, where possibly $K_{\mathbb{Z}} \neq 8$. Namely, one finds that

$$\mathsf{M}_p(n) = O(n \lg p \lg(n \lg p)\, K_{\mathbb{F}}^{\max(0,\log^* n - \log^* p)} K_{\mathbb{Z}}^{\log^* p}) \tag{1.5}$$

where $K_{\mathbb{F}} = 8$ (we omit the proof). Our Theorem 1.1 asserts that (1.5) holds with $K_{\mathbb{F}} = 4$.

1.3. **Overview of the new algorithm.** To explain the new approach, let us first recall the idea behind the polynomial multiplication algorithm of [15].

Consider a polynomial multiplication problem in $\mathbb{F}_p[X]$, where the degree n is very large compared to p. By splitting the inputs into chunks, we convert this to a bivariate multiplication problem in $\mathbb{F}_p[Y, Z]/(f(Y), Z^m - 1)$, for a suitable integer m and irreducible polynomial $f \in \mathbb{F}_p[Y]$. This bivariate product is handled by means of DFTs (discrete Fourier transforms) of length m over $\mathbb{F}_p[Y]/f$. The key innovation of [15] was to choose $\deg f$ so that $p^{\deg f} - 1$ is divisible by many small primes, so many, in fact, that their product is comparable to n, even though $\deg f$ itself is exponentially smaller. This is possible thanks to a number-theoretic result of Adleman, Pomerance and Rumely [1], building on earlier work of Prachar [17]. Taking m to be a product of many of these primes, we obtain $m \mid p^{\deg f} - 1$, and hence $\mathbb{F}_p[Y]/f$ contains a root of unity of order m. As m is highly composite, each DFT of length m may be converted to a collection of much smaller DFTs via the Cooley–Tukey method. These in turn are converted into multiplication problems using Bluestein's algorithm. These multiplications, corresponding to exponentially smaller values of n, are handled recursively.

The recursion continues until n becomes comparable to p. The number of recursion levels during this phase is $\log^* n - \log^* p + O(1)$, and the constant $K_{\mathbb{F}} = 8$ represents the "expansion factor" that occurs at each recursion level, due to phenomena such as zero-padding. When n becomes comparable to p, the algorithm switches strategy to Kronecker substitution combined with ordinary integer multiplication. This phase contributes the $K_{\mathbb{Z}}^{\log^* p}$ term.

It was pointed out in [13, §8] that the value of $K_{\mathbb{F}}$ can be improved to $K_{\mathbb{F}} = 4$ if one is willing to accept certain unproved number-theoretic conjectures, including Artin's conjecture on primitive roots. More precisely, under these conjectures, one may find an irreducible f of the form $f(Y) = Y^{\alpha-1} + \cdots + Y + 1$, where α is prime, so that $\mathbb{F}_p[Y, Z]/(f(Y), Z^m - 1)$ is a direct summand of $\mathbb{F}_p[Y, Z]/(Y^\alpha - 1, Z^m - 1)$. This last ring is isomorphic to $\mathbb{F}_p[X]/(X^{\alpha m} - 1)$, and one may use this isomorphism to save a factor of two in zero-padding at each recursion level. These savings lead directly to the improved value for $K_{\mathbb{F}}$.

To prove Theorem 1.1, we will pursue a variant of this idea. We will take f to be a cyclotomic polynomial $\phi_\alpha(Y)$ for a judiciously chosen integer α (not necessarily prime). Since $\phi_\alpha \mid Y^\alpha - 1$, we may use the above isomorphism to realise the same economy in zero-padding as in the conjectural construction of [13, §8]. However, unlike [13], we do not require that f be irreducible in $\mathbb{F}_p[Y]$. Thus $\mathbb{F}_p[Y]/f$ is no longer in general a field, but a direct sum of fields. The situation is reminiscent of Fürer's algorithm, in which the coefficient ring $\mathbb{C}[Y]/(Y^{2^r} + 1)$ is not a field, but a

direct sum of copies of $\mathbb{C}$. The key technical contribution of this paper is to show that we have enough control over the factorisation of ϕ_α in $\mathbb{F}_p[Y]$ to ensure that $\mathbb{F}_p[Y]/\phi_\alpha$ contains suitable principal roots of unity. This approach avoids Artin's conjecture and other number-theoretic difficulties, and enables us to reach $K_{\mathbb{F}} = 4$ unconditionally. The construction of α is the subject of Section 3, and the main polynomial multiplication algorithm is presented in Section 4.

2. Preliminaries

2.1. **Logarithmically slow functions.** Let $x_0 \in \mathbb{R}$, and let $\Phi : (x_0, \infty) \to \mathbb{R}$ be a smooth increasing function. We recall from [14, §5] that Φ is said to be *logarithmically slow* if there exists an integer $\ell \geqslant 0$ such that

$$(\log^{\circ \ell} \circ \Phi \circ \exp^{\circ \ell})(x) = \log x + O(1)$$

as $x \to \infty$. For example, the functions $\log(5x)$, $5 \log x$, $(\log x)^5$, and $2^{(\log \log x)^5}$ are logarithmically slow, with $\ell = 0, 1, 2, 3$ respectively.

We will always assume that x_0 is chosen large enough to ensure that $\Phi(x) \leqslant x - 1$ for all $x > x_0$. According to [14, Lemma 2], this is possible for any logarithmically slow function, and it implies that the iterator $\Phi^*(x) := \min\{k \geqslant 0 : \Phi^{\circ k}(x) \leqslant x_0\}$ is well-defined on $\mathbb{R}$. It is shown in [14, Lemma 3] that this iterator satisfies

$$\Phi^*(x) = \log^* x + O(1) \tag{2.1}$$

as $x \to \infty$. In other words, logarithmically slow functions are more or less indistinguishable from $\log x$, as far as iterators are concerned.

As in [14] and [15], we will use logarithmically slow functions to measure *size reduction* in multiplication algorithms. The typical situation is that we have a function $T(n)$ measuring the (normalised) cost of a certain multiplication algorithm for inputs of size n; we reduce the problem to a collection of problems of size $n_i < \Phi(n)$, leading to a bound for $T(n)$ in terms of the various $T(n_i)$. Applying the reduction recursively, we wish to convert these bounds into an explicit asymptotic estimate for $T(n)$. For this purpose we recall the following 'master theorem' [14, Prop. 8].

Proposition 2.1. *Let $K > 1$, $B \geqslant 0$, and let $\ell \geqslant 0$ be an integer. Let $x_0 \geqslant \exp^{\circ \ell}(1)$, and let $\Phi : (x_0, \infty) \to \mathbb{R}$ be a logarithmically slow function such that $\Phi(x) \leqslant x - 1$ for all $x > x_0$. Then there exists a positive constant C (depending on x_0, Φ, K, B and ℓ) with the following property.*

Let $\sigma \geqslant x_0$ and $L > 0$. Let $\mathcal{S} \subseteq \mathbb{R}$, and let $T : \mathcal{S} \to \mathbb{R}^{\geqslant}$ be any function satisfying the following recurrence. First, $T(y) \leqslant L$ for all $y \in \mathcal{S}$, $y \leqslant \sigma$. Second, for all $y \in \mathcal{S}$, $y > \sigma$, there exist $y_1, \ldots, y_d \in \mathcal{S}$ with $y_i \leqslant \Phi(y)$, and weights $\gamma_1, \ldots, \gamma_d \geqslant 0$ with $\sum_i \gamma_i = 1$, such that

$$T(y) \leqslant K \left(1 + \frac{B}{\log^{\circ \ell} y}\right) \sum_{i=1}^{d} \gamma_i T(y_i) + L.$$

Then for all $y \in \mathcal{S}$, $y > \sigma$, we have

$$T(y) \leqslant C L K^{\log^* y - \log^* \sigma}.$$

2.2. **Discrete Fourier transforms.** Let $n \geqslant 1$ and let $\mathcal{R}$ be a commutative ring in which n is invertible. A *principal n-th root of unity* is an element $\omega \in \mathcal{R}$ such that $\omega^n = 1$ and such that $\sum_{j=0}^{n-1} \omega^{ij} = 0$ for $i = 1, 2, \ldots, n-1$. If m is a divisor of n, then $\omega^{n/m}$ is easily seen to be a principal m-th root of unity.

Fix a principal n-th root of unity ω. The *discrete Fourier transform* (DFT) of the sequence $(a_0, \ldots, a_{n-1}) \in \mathcal{R}^n$ with respect to ω is the sequence $(\hat{a}_0, \ldots, \hat{a}_{n-1}) \in \mathcal{R}^n$ defined by $\hat{a}_j := \sum_{i=0}^{n-1} \omega^{ij} a_i$. Equivalently, $\hat{a}_j = A(\omega^j)$ where $A = \sum_{i=0}^{n-1} a_i X^i \in \mathcal{R}[X]/(X^n - 1)$.

The *inverse DFT* recovers $(a_0, \ldots, a_{n-1})$ from $(\hat{a}_0, \ldots, \hat{a}_{n-1})$. Computationally it corresponds to a DFT with respect to ω^{-1}, followed by a division by n, because

$$\frac{1}{n} \sum_{j=0}^{n-1} \omega^{-kj} \hat{a}_j = \frac{1}{n} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \omega^{(i-k)j} a_i = a_k, \qquad k = 0, \ldots, n-1.$$

DFTs may be used to implement cyclic convolutions. Suppose that we wish to compute $C := AB$ where $A, B \in \mathcal{R}[X]/(X^n - 1)$. We first perform DFTs to compute $A(\omega^i)$ and $B(\omega^i)$ for $i = 0, \ldots, n-1$. We then compute $C(\omega^i) = A(\omega^i)B(\omega^i)$ for each i, and finally perform an inverse DFT to recover $C \in \mathcal{R}[X]/(X^n - 1)$.

This strategy may be generalised to handle a multidimensional cyclic convolution, that is, to compute $C := AB$ for

$$A, B \in \mathcal{R}[X_1, \ldots, X_d]/(X_1^{n_1} - 1, \ldots, X_d^{n_d} - 1).$$

For this, we require that each n_k be invertible in $\mathcal{R}$, and that $\mathcal{R}$ contain a principal n_k-th root of unity ω_k for each k. Let $n := n_1 \cdots n_d$. We first perform multidimensional DFTs to evaluate A and B at the n points $\{(\omega_1^{j_1}, \ldots, \omega_d^{j_d}) : 0 \leqslant j_k < n_k\}$. We then multiply pointwise, and finally recover C via a multidimensional inverse DFT.

Each multidimensional DFT may be reduced to a collection of one-dimensional DFTs as follows. We first compute $A(X_1, \ldots, X_{d-1}, \omega_d^j) \in \mathcal{R}[X_1, \ldots, X_{d-1}]$ for each $j = 0, \ldots, n_d - 1$; this involves n/n_d DFTs of length n_d. We then recursively evaluate each of these polynomials at the n/n_d points $(\omega_1^{j_1}, \ldots, \omega_{d-1}^{j_{d-1}})$. Altogether, this strategy involves computing n/n_k DFTs of length n_k for each $k = 1, \ldots, d$.

Finally, we briefly recall Bluestein's method [4] for reducing a (one-dimensional) DFT to a convolution problem (see also [14, §2.5]). Let $n \geqslant 1$ be odd and let $\omega \in \mathcal{R}$ be a principal n-th root of unity. Set $\xi := \omega^{(n+1)/2}$, so that $\xi^2 = \omega$ and $\xi^n = 1$. Then computing the DFT of a given sequence $(a_0, \ldots, a_{n-1}) \in \mathcal{R}^n$ with respect to ω reduces to computing the product of the polynomials

$$f(Z) := \sum_{i=0}^{n-1} \xi^{i^2} a_i Z^i, \qquad g(Z) := \sum_{i=0}^{n-1} \xi^{-i^2} Z^i$$

in $\mathcal{R}[Z]/(Z^n - 1)$, plus $O(n)$ auxiliary multiplications in $\mathcal{R}$. Notice that $g(Z)$ is fixed and does not depend on the input sequence.

2.3. **Data layout.** In this section we discuss several issues relating to the layout of data on the Turing machine tapes.

Integers will always be stored in the standard binary representation. If n is a positive integer, then elements of $\mathbb{Z}/n\mathbb{Z}$ will always be stored as residues in the range $0 \leqslant x < n$, occupying $\lg n$ bits of storage.

If $\mathcal{R}$ is a ring and $f \in \mathcal{R}[X]$ is a polynomial of degree $n \geqslant 1$, then an element of $\mathcal{R}[X]/f(X)$ will always be represented as a sequence of n coefficients in the standard monomial order. This convention is applied recursively, so for rings of the type $(\mathcal{R}[Y]/f(Y))[X]/g(X)$, the coefficient of X^0 is stored first, as an element of $\mathcal{R}[Y]/f(Y)$, followed by the coefficient of X^1, and so on.

A multidimensional array of size $n_d \times \cdots \times n_1$, whose entries occupy b bits each, will be stored as a linear array of $bn_1 \cdots n_d$ bits. The entries are ordered lexicographically in the order $(0, \ldots, 0, 0), (0, \ldots, 0, 1), \ldots, (n_d - 1, \ldots, n_1 - 1)$. In particular, an element of $(\cdots(\mathcal{R}[X_1]/f_1(X_1))\cdots)[X_d]/f_d(X_d)$ is represented as an $n_d \times \cdots \times n_1$ array of elements of $\mathcal{R}$. We will generally prefer the more compact notation $\mathcal{R}[X_1, \ldots, X_d]/(f_1(X_1), \ldots, f_d(X_d))$.

There are many instances where an $n \times m$ array must be transposed so that its entries can be accessed efficiently either 'by columns' or 'by rows'. Using the algorithm of [5, Lemma 18], such a transposition may be achieved in $O(bnm \lg \min(n, m))$ bit operations, where b is the bit size of each entry. (The idea of the algorithm is to split the array in half along the short dimension, and transpose each half recursively.)

One important application is the following result, which estimates the data rearrangement cost associated to the the Agarwal–Cooley method [2] for converting between one-dimensional and multidimensional convolution problems (this is closely related to the Good–Thomas DFT algorithm [10, 22]).

Lemma 2.2. *Let $n, m \geqslant 2$ be relatively prime, and let $\mathcal{R}$ be a ring whose elements are represented using b bits. There exists an isomorphism*

$$\mathcal{R}[X]/(X^{nm} - 1) \cong \mathcal{R}[Y, Z]/(Y^n - 1, Z^m - 1)$$

that may be evaluated in either direction in $O(bnm \lg \min(n, m))$ bit operations.

Proof. Let $c := m^{-1} \bmod n$, and let

$$\beta : \mathcal{R}[X]/(X^{nm} - 1) \to \mathcal{R}[Y, Z]/(Y^n - 1, Z^m - 1)$$

denote the homomorphism that maps X to $Y^c Z$, and acts as the identity on $\mathcal{R}$. Suppose that we wish to compute $\beta(F)$ for some input polynomial $F = \sum_{k=0}^{nm-1} F_k X^k \in \mathcal{R}[X]/(X^{nm} - 1)$. Interpreting the list $(F_0, \ldots, F_{nm-1})$ as an $n \times m$ array, the (i, j)-th entry corresponds to F_{im+j}. After transposing the array, which costs $O(bnm \lg \min(n, m))$ bit operations, we have an $m \times n$ array, whose (j, i)-th entry is F_{im+j}. Now for each j, cyclically permute the j-th row by $(jc \bmod n)$ slots; altogether this uses only $O(bnm)$ bit operations. The result is an $m \times n$ array whose (j, i)-th entry is $F_{(i-jc \bmod n)m+j}$, which is exactly the coefficient of $Y^{((i-jc)m+j)c} Z^{(i-jc)m+j} = Y^i Z^j$ in $\beta(F)$. The inverse map β^{-1} may be computed by reversing this procedure. □

Corollary 2.3. *Let $n_1, \ldots, n_d \geqslant 2$ be pairwise relatively prime, let $n := n_1 \cdots n_d$, and let $\mathcal{R}$ be a ring whose elements are represented using b bits. There exists an isomorphism*

$$\mathcal{R}[X]/(X^n - 1) \cong \mathcal{R}[X_1, \ldots, X_d]/(X_1^{n_1} - 1, \ldots, X_d^{n_d} - 1)$$

that may be evaluated in either direction in $O(bn \lg n)$ bit operations.

Proof. Using Lemma 2.2, we may construct a sequence of isomorphisms

$$\begin{aligned}\mathcal{R}[X]/(X^{n_1\cdots n_d}-1) &\cong \mathcal{R}[X_1,W_2]/(X_1^{n_1}-1,W_2^{n_2\cdots n_d}-1)\\ &\cong \mathcal{R}[X_1,X_2,W_3]/(X_1^{n_1}-1,X_2^{n_2}-1,W_3^{n_3\cdots n_d}-1)\\ &\cdots\\ &\cong \mathcal{R}[X_1,\ldots,X_d]/(X_1^{n_1}-1,\ldots,X_d^{n_d}-1),\end{aligned}$$

the i-th of which may be computed in $O(bn\lg n_i)$ bit operations. The overall cost is $O(\sum_i bn\lg n_i) = O(bn\lg n)$ bit operations. □

3. Cyclotomic coefficient rings

The aim of this section is to construct certain coefficient rings that play a central role in the multiplication algorithms described later. The basic idea is as follows. Suppose that we want to multiply two polynomials in $\mathbb{F}_p[X]$, and that the degree of the product is known to be at most n. If N is an integer with $N > n$, then by appropriate zero-padding, we may embed the problem in $\mathbb{F}_p[X]/(X^N-1)$. Furthermore, if we have some factorisation $N = \alpha m$, where α and m are relatively prime, then there is an isomorphism

$$\mathbb{F}_p[X]/(X^N-1) \cong \mathbb{F}_p[Y,Z]/(Y^\alpha-1, Z^m-1),$$

and the latter ring is closely related to

$$\mathbb{F}_p[Y,Z]/(\phi_\alpha(Y), Z^m-1) \cong (\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m-1)$$

(recall that $\phi_\alpha(Y)$ denotes the α-th cyclotomic polynomial). In particular, computing the product in $(\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m-1)$ recovers 'most' of the information about the product in $\mathbb{F}_p[X]/(X^N-1)$.

In this section we show how to choose N, α and m with the following properties:

(1) N is not much larger than n, so that not too much space is 'wasted' in the initial zero-padding step;
(2) $\varphi(\alpha)$ $(= \deg\phi_\alpha)$ is not much smaller than α, so that we do not lose much information by working modulo $\phi_\alpha(Y)$ instead of modulo $Y^\alpha - 1$ (this missing information must be recovered by other means);
(3) the coefficient ring $\mathbb{F}_p[Y]/\phi_\alpha$ contains a principal m-th root of unity, so that we can multiply in $(\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m-1)$ efficiently by means of DFTs over $\mathbb{F}_p[Y]/\phi_\alpha$;
(4) m is a product of many integers that are exponentially smaller than n, so that the DFTs of length m may be reduced to many small DFTs; and
(5) α is itself exponentially smaller than n.

The last two items ensure that the small DFTs can be converted to multiplication problems of degree exponentially smaller than n, to allow the recursion to proceed.

Definition 3.1. An *admissible tuple* is a sequence $(q_0, q_1, \ldots, q_e)$ of distinct primes $(e \geqslant 1)$ satisfying the following conditions. First,

$$(\lg N)^3 < q_i < 2^{(\lg\lg N)^2}, \qquad i = 0,\ldots,e, \tag{3.1}$$

where $N := q_0\cdots q_e$. Second, $q_i - 1$ is squarefree for $i = 1,\ldots,e$, and

$$\lambda(q_0,\ldots,q_e) := \mathrm{LCM}(q_1-1,\ldots,q_e-1) < 2^{(\lg\lg N)^2}. \tag{3.2}$$

(Note that $q_0 - 1$ need not be squarefree, and q_0 does not participate in (3.2).)

An *admissible length* is a positive integer N of the form $N = q_0 \cdots q_e$ where $(q_0, \ldots, q_e)$ is an admissible tuple.

If N is an admissible length, we treat $(q_0, \ldots, q_e)$ and $\lambda(N) := \lambda(q_0, \ldots, q_e)$ as auxiliary data attached to N. For example, if an algorithm takes N as input, we implicitly assume that this auxiliary data is also supplied as part of the input.

Example 3.2. Let $n = 10^{100000}$. There is an admissible length N, slightly larger than n, given by

$$\begin{aligned} N &= 10000000000000000000156121 \ldots (99971 \text{ digits omitted}) \ldots 26353 \\ &= q_0 q_1 \cdots q_{6035} \end{aligned}$$

where

$$\begin{aligned} q_0 &= 206658761261792645783, \\ q_1 &= 36658226833235899 = 1 + 2 \cdot 3 \cdot 11 \cdot 17 \cdot 23 \cdot 29 \cdot 37 \cdot 53 \cdot 59 \cdot 67 \cdot 71 \cdot 89, \\ q_2 &= 36658244723486119 = 1 + 2 \cdot 3 \cdot 17 \cdot 29 \cdot 47 \cdot 59 \cdot 67 \cdot 73 \cdot 83 \cdot 101 \cdot 109, \\ q_3 &= 36658319675739343 = 1 + 2 \cdot 3 \cdot 7 \cdot 17 \cdot 29 \cdot 31 \cdot 41 \cdot 47 \cdot 53 \cdot 61 \cdot 89 \cdot 103, \\ q_4 &= 36658428883190467 = 1 + 2 \cdot 3 \cdot 11 \cdot 31 \cdot 43 \cdot 61 \cdot 71 \cdot 73 \cdot 107 \cdot 109 \cdot 113, \\ &\cdots \\ q_{6035} &= 37076481100386859 = 1 + 2 \cdot 3 \cdot 13 \cdot 29 \cdot 31 \cdot 59 \cdot 83 \cdot 97 \cdot 101 \cdot 103 \cdot 107 \end{aligned}$$

and

$$\lambda(N) = 2 \cdot 3 \cdot 5 \cdots 113 = 3161005464041760778814520629154366249327468699 0.$$

Definition 3.3. Let p be a prime. An admissible length N is called *p-admissible* if $N > p^2$ and $p \nmid N$ (i.e., p is distinct from $q_0, \ldots, q_e$).

The following result explains how to choose a p-admissible length close to any prescribed target.

Proposition 3.4. *There is an absolute constant $z_1 > 0$ with the following property. Given as input a prime p and an integer $n > \max(z_1, p^2)$, in $2^{O((\lg \lg n)^2)}$ bit operations we may compute a p-admissible length N in the interval*

$$n < N < \left(1 + \frac{1}{\lg n}\right) n. \tag{3.3}$$

The key ingredient in the proof is the following number-theoretic result of Adleman, Pomerance and Rumely.

Lemma 3.5 ([1, Prop. 10])**.** *There is an absolute constant $C_1 > 0$ with the following property. For all $x > 10$, there exists a positive squarefree integer $\lambda_0 < x^2$ such that*

$$\sum_{\substack{q \text{ prime} \\ q-1 \mid \lambda_0}} 1 > \exp(C_1 \log x / \log \log x).$$

Proof of Proposition 3.4. Let $\lambda_{\max} := \lceil 2^{\frac{2}{9}(\lg \lg n)^2} \rceil$, and for $\lambda \geqslant 1$ define $f(\lambda)$ to be the number of primes q in the interval $(\lg n)^4 < q \leqslant \lambda_{\max} + 1$ such that $q - 1 \mid \lambda$ and $q \neq p$. We claim that, provided n is large enough, there exists some squarefree $\lambda_0 \in \{1, \ldots, \lambda_{\max}\}$ such that $f(\lambda_0) > \lg n$. To see this, apply Lemma 3.5 with $x := 2^{\frac{1}{9}(\lg \lg n)^2}$; for large n we then have

$$C_1 \log x / \log \log x > 15 (\log_2 x)^{1/2} = 5 \lg \lg n,$$

so Lemma 3.5 implies that there exists a positive squarefree integer $\lambda_0 < x^2 \leqslant \lambda_{\max}$ for which

$$\sum_{\substack{q \text{ prime} \\ q-1 \mid \lambda_0}} 1 > \exp(5 \lg\lg n) > (\lg n)^5$$

and hence

$$f(\lambda_0) = \sum_{\substack{(\lg n)^4 < q \leqslant \lambda_{\max}+1 \\ q \text{ prime},\ q \neq p \\ q-1 \mid \lambda_0}} 1 > (\lg n)^5 - (\lg n)^4 - 1 > \lg n.$$

We may locate one such λ_0 by means of the following algorithm (adapted from the proof of [15, Lemma 4.5]). First use a sieve to enumerate the primes q in the interval $(\lg n)^4 < q \leqslant \lambda_{\max} + 1$, and to determine which $\lambda = 1, \ldots, \lambda_{\max}$ are squarefree, in $(\lambda_{\max})^{1+o(1)}$ bit operations. Now initialise an array of integers $c_\lambda := 0$ for $\lambda = 1, \ldots, \lambda_{\max}$. For each $q \neq p$, scan through the array, incrementing those c_λ for which λ is squarefree and divisible by $q - 1$, and stop as soon as one of the c_λ reaches $\lg n$. We need only allocate $O(\lg\lg n)$ bits per array entry, so each pass through the array costs $O(\lambda_{\max} \lg\lg n)$ bit operations. The number of passes is $O(\lambda_{\max})$, so the total cost of finding a suitable λ_0 is $O(\lambda_{\max}^2 \lg\lg n) = 2^{O((\lg\lg n)^2)}$ bit operations. Within the same time bound, we may also easily recover a list of primes $q_1, q_2, \ldots, q_{\lg n}$ for which $q_i - 1 \mid \lambda_0$.

Next, compute the partial products q_1, $q_1 q_2$, …, $q_1 q_2 \cdots q_{\lg n}$, and determine the smallest integer $e \geqslant 1$ for which $q_1 \cdots q_e > n/2^{\frac{1}{2}(\lg\lg n)^2}$. Such an e certainly exists, as $q_1 \cdots q_{\lg n} \geqslant 2^{\lg n} \geqslant n$. Since each q_i occupies $O((\lg\lg n)^2)$ bits, this can all be done in $(\lg n)^{O(1)}$ bit operations. Also, as

$$q_e \leqslant \lambda_0 + 1 \leqslant 2^{\frac{2}{9}(\lg\lg n)^2} + 1 < 2^{\frac{1}{4}(\lg\lg n)^2}$$

and $q_1 \cdots q_{e-1} \leqslant n/2^{\frac{1}{2}(\lg\lg n)^2}$, we find that

$$2^{\frac{1}{4}(\lg\lg n)^2} < \frac{n}{q_1 \cdots q_e} < 2^{\frac{1}{2}(\lg\lg n)^2}$$

for large n.

Let q_0 be the least prime that exceeds $n/(q_1 \cdots q_e)$ and that is distinct from p. According to [3], the interval $[x - x^{0.525}, x]$ contains at least one prime for all sufficiently large x; therefore

$$\begin{aligned} q_0 &< \frac{n}{q_1 \cdots q_e} + \left(\frac{n}{q_1 \cdots q_e}\right)^{0.6} \\ &< \left(1 + (2^{\frac{1}{4}(\lg\lg n)^2})^{-0.4}\right) \frac{n}{q_1 \cdots q_e} < \left(1 + \frac{1}{\lg n}\right) \frac{n}{q_1 \cdots q_e} \end{aligned}$$

for n sufficiently large. We may find q_0 in $2^{O((\lg\lg n)^2)}$ bit operations, by using trial division to test successive integers for primality.

Set $N := q_0 q_1 \cdots q_e$. Then (3.3) holds, and certainly $N > p^2$ and $p \nmid N$. Let us check that $(q_0, \ldots, q_e)$ is admissible, provided n is large enough. For $i = 1, \ldots, e$ we have

$$(\lg N)^3 < (\lg n)^4 < q_i \leqslant \lambda_0 + 1 < 2^{\frac{1}{4}(\lg\lg n)^2} < 2^{(\lg\lg N)^2},$$

and also

$$(\lg N)^3 < 2^{\frac{1}{4}(\lg\lg n)^2} < q_0 < \left(1 + \frac{1}{\lg n}\right) 2^{\frac{1}{2}(\lg\lg n)^2} < 2^{(\lg\lg N)^2};$$

this establishes (3.1). Also, as $q_0 > 2^{\frac{1}{4}(\lg\lg n)^2} > q_i$ for $i = 1, \ldots, e$, we see that q_0 is distinct from $q_1, \ldots, q_e$. Finally, (3.2) holds because

$$\mathrm{LCM}(q_1 - 1, \ldots, q_e - 1) \mid \lambda_0 \leqslant 2^{\frac{2}{9}(\lg\lg n)^2} < 2^{(\lg\lg N)^2}.$$

This also shows that we may compute the auxiliary data $\lambda(q_0, \ldots, q_e)$ in $2^{O((\lg\lg n)^2)}$ bit operations. □

Remark 3.6. Example 3.2 was constructed by enumerating the smallest primes $q_1, q_2, \ldots$ exceeding $(\lg n)^3$ for which $q_i - 1 \mid 2 \cdot 3 \cdot 5 \cdots 113$, halting just before their product reached n, and then choosing q_0 to make N as close to n as possible. The proof of Proposition 3.4 goes a different way: rather than choosing λ first, the proof constructs $q_1, \ldots, q_e$ and λ simultaneously. In particular, one cannot guarantee that λ will be a product of an initial segment of primes, as occurred in the example. Indeed, the proof of [1, Prop. 10] (and of its predecessor [17]) yields very little information at all about the prime factorisation of λ. For further discussion, see [1, Remark 6.2].

Definition 3.7. Let p be a prime and let $N = q_0 \cdots q_e$ be a p-admissible length. A *p-admissible divisor* of N is a positive divisor α of N, with $q_0 \mid \alpha$, such that the ring $\mathbb{F}_p[Y]/\phi_\alpha(Y)$ contains a principal $(q_1 \cdots q_e)$-th root of unity, and such that

$$\lg N < \alpha < 2^{(\lg\lg N)^4} \tag{3.4}$$

and

$$\varphi(\alpha) > \left(1 - \frac{1}{\lg N}\right) \alpha. \tag{3.5}$$

The next result shows how to construct a p-admissible divisor for any sufficiently large p-admissible length N. The idea behind the construction is as follows. Let $\mathrm{ord}_n\, p$ denote the order of p in the multiplicative group of integers modulo n. For any $\alpha \geqslant 1$, not divisible by p, the ring $\mathbb{F}_p[Y]/\phi_\alpha(Y)$ is a direct sum of fields of order p^r, where $r = \mathrm{ord}_\alpha\, p$ [23, Lemma 14.50]. Our goal is to ensure that $p^r - 1$ is divisible by $q_1 \cdots q_e$, so that $\mathbb{F}_p[Y]/\phi_\alpha(Y)$ contains the desired principal root of unity. One way to force q_i to divide $p^r - 1$ is simply to choose α divisible by q_i, as this implies that $\mathrm{ord}_{q_i}\, p \mid r$. The difficulty is that we cannot do this for *all* q_i, because then α would become too large, violating (3.4). Fortunately, we can take advantage of the fact that the $q_i - 1$ share a small common multiple $\lambda = \lambda(N)$; this enables us to take α to be a product of a *small* subset of the q_i, in such a way that still every one of $q_1, \ldots, q_e$ divides $p^r - 1$.

Proposition 3.8. *There is an absolute constant $z_2 > 0$ with the following property. Given as input a prime p and a p-admissible length $N > z_2$, we may compute a p-admissible divisor α of N, together with the cyclotomic polynomial $\phi_\alpha \in \mathbb{F}_p[Y]$ and a principal $(q_1 \cdots q_e)$-th root of unity in $\mathbb{F}_p[Y]/\phi_\alpha$, in $2^{O((\lg\lg N)^4)} p^{1+o(1)}$ bit operations.*

Proof. We are given as input an admissible tuple $(q_0, \ldots, q_e)$ with $N = q_0 \cdots q_e$, and the squarefree integer $\lambda := \lambda(q_0, \ldots, q_e)$. Let $\mathcal{L}$ be the set of primes dividing λ.

By (3.2) we have $|\mathcal{L}| \leqslant \log_2 \lambda < (\lg\lg N)^2$, and we may compute $\mathcal{L}$ in $\lambda^{O(1)} = 2^{O((\lg\lg N)^2)}$ bit operations.

We start by computing a table of values of $\operatorname{ord}_{q_i} p$ for $i = 1, \ldots, e$; note that $p \neq q_i$ by hypothesis, so $\operatorname{ord}_{q_i} p$ is well-defined. We have $q_i - 1 \mid \lambda$ and hence $\operatorname{ord}_{q_i} p \mid \lambda$ for each i. To compute $\operatorname{ord}_{q_i} p$, we first compute $p \bmod q_i$ in $O(\lg q_i \lg p)$ bit operations, and then repeatedly multiply by p modulo q_i until reaching 1. Since $\operatorname{ord}_{q_i} p \leqslant \lambda$, and there are $e = O(\lg N)$ primes q_i, the total cost to compute the table is

$$O((\lambda \lg^2 q_i + \lg q_i \lg p) \lg N) = (2^{(\lg\lg N)^2} \lg p)^{O(1)}$$

bit operations.

Using the above table, we construct a certain vector $\sigma = (\sigma_1, \ldots, \sigma_e) \in \{0,1\}^e$ as follows. Initialise the vector as $\sigma := (0, \ldots, 0)$. For each $\ell \in \mathcal{L}$, search for the smallest $i = 1, \ldots, e$ such that $\ell \mid \operatorname{ord}_{q_i} p$. If such an i is found, set $\sigma_i := 1$; if no i is found, ignore this ℓ. The cost of computing σ is $O(|\mathcal{L}| e (\lg\lambda)^2) = (\lg N)^{O(1)}$ bit operations.

Set $\alpha := q_0 \prod_{i:\sigma_i=1} q_i$. To establish (3.4), note that the number of i for which $\sigma_i = 1$ is at most $|\mathcal{L}|$, so (3.1) implies that

$$\lg N < q_0 \leqslant \alpha < (2^{(\lg\lg N)^2})^{|\mathcal{L}|+1} \leqslant (2^{(\lg\lg N)^2})^{(\lg\lg N)^2} = 2^{(\lg\lg N)^4}.$$

For (3.5), first observe that

$$\frac{\varphi(\alpha)}{\alpha} = \left(1 - \frac{1}{q_0}\right) \prod_{i:\sigma_i=1} \left(1 - \frac{1}{q_i}\right) > \left(1 - \frac{1}{(\lg N)^3}\right)^{(\lg\lg N)^2}.$$

Since $-\log(1-\varepsilon) < 2\varepsilon$ for any $\varepsilon \in (0, \frac{1}{2})$, we obtain

$$-\log \frac{\varphi(\alpha)}{\alpha} < \frac{2(\lg\lg N)^2}{(\lg N)^3} < \frac{1}{\lg N}$$

and hence $\varphi(\alpha)/\alpha > \exp(-1/\lg N) > 1 - 1/\lg N$ for sufficiently large N.

Now compute the cyclotomic polynomial $\phi_\alpha \in \mathbb{F}_p[Y]$ (i.e., the reduction modulo p of $\phi_\alpha(Y) \in \mathbb{Z}[Y]$). This can be done in $(\alpha \lg p)^{O(1)}$ bit operations, using for example [23, Algorithm 14.48]. We may then determine the factorisation of ϕ_α into irreducibles in $\mathbb{F}_p[Y]$, say $\phi_\alpha = f_1 \cdots f_k$, in $\alpha^{O(1)} p^{1/2+o(1)}$ bit operations [20, Thm. 1]. Since $p \nmid \alpha$, the f_j are distinct, and each f_j has degree $r := \operatorname{ord}_\alpha p$ [23, Lemma 14.50]. In other words, $\mathbb{F}_p[Y]/\phi_\alpha$ is isomorphic to a direct sum of k copies of $\mathbb{F}_{p^r}$.

We claim that $q_h \mid p^r - 1$ for all $h = 1, \ldots, e$. For this, it suffices to prove that $\operatorname{ord}_{q_h} p \mid r$ for each h. Since λ is squarefree, it suffices in turn to show that every prime ℓ dividing $\operatorname{ord}_{q_h} p$ also divides r. But for every such ℓ, the procedure for constructing σ must have succeeded in finding *some* i for which $\ell \mid \operatorname{ord}_{q_i} p$ (since at least one value of i works, namely $i = h$). Then $\sigma_i = 1$ for this i, so $q_i \mid \alpha$. This implies that $\operatorname{ord}_{q_i} p \mid \operatorname{ord}_\alpha p = r$, and hence that $\ell \mid r$.

We conclude that $q_1 \cdots q_e \mid p^r - 1$, so each $\mathbb{F}_p[Y]/f_j$ contains a primitive root of unity of order $q_1 \cdots q_e$. As the factorisation of $q_1 \cdots q_e$ is known, we may locate one such primitive root in each $\mathbb{F}_p[Y]/f_j$ in $\alpha^{O(1)} p^{1+o(1)}$ bit operations [21] (see also [14, Lemma 3.3]). Combining these primitive roots via the Chinese remainder theorem, we obtain the desired principal $(q_1 \cdots q_e)$-th root of unity in $\mathbb{F}_p[Y]/\phi_\alpha$ in another $(\alpha \lg p)^{O(1)}$ bit operations. □

Remark 3.9. The $p^{1+o(1)}$ term in Proposition 3.8 arises from the best known deterministic complexity bounds for factoring polynomials and finding primitive roots. If we permit randomised algorithms, then $p^{1+o(1)}$ may be replaced by $(\lg p)^{O(1)}$. This has no effect on the main results of this paper.

Example 3.10. Continuing with Example 3.2, let us take $p = 3$. In the notation of the proof of Proposition 3.8, we have $\mathcal{L} = \{2, 3, 5, \dots, 113\}$. For each $\ell \in \mathcal{L}$, let us write $q^{(\ell)}$ for the smallest q_i for which $\operatorname{ord}_{q_i} 3$ is divisible by ℓ. Then we have

$$\begin{aligned}
q^{(2)} &= q_1, & q^{(3)} &= q_1, & q^{(5)} &= q_5, & q^{(7)} &= q_9, & q^{(11)} &= q_1,\\
q^{(13)} &= q_5, & q^{(17)} &= q_1, & q^{(19)} &= q_5, & q^{(23)} &= q_1, & q^{(29)} &= q_1,\\
q^{(31)} &= q_3, & q^{(37)} &= q_1, & q^{(41)} &= q_3, & q^{(43)} &= q_4, & q^{(47)} &= q_2,\\
q^{(53)} &= q_1, & q^{(59)} &= q_1, & q^{(61)} &= q_3, & q^{(67)} &= q_1, & q^{(71)} &= q_1,\\
q^{(73)} &= q_2, & q^{(79)} &= q_6, & q^{(83)} &= q_2, & q^{(89)} &= q_1, & q^{(97)} &= q_5,\\
q^{(101)} &= q_2, & q^{(103)} &= q_3, & q^{(107)} &= q_4, & q^{(109)} &= q_2, & q^{(113)} &= q_4.
\end{aligned}$$

Therefore $\sigma_i = 1$ for $i = 1, 2, 3, 4, 5, 6, 9$, and we have

$$\begin{aligned}
\alpha &= q_0 q_1 q_2 q_3 q_4 q_5 q_6 q_9\\
&\approx 1.8385309928916569681 \times 10^{136},\\
\varphi(\alpha) &\approx 1.8385309928916566171 \times 10^{136},\\
r &= \operatorname{ord}_\alpha 3 = 2 \cdot 3 \cdot 5 \cdots 109 \cdot 113 \cdot 883 \cdot 9041 \cdot 327251 \cdot 39551747.
\end{aligned}$$

The ring $\mathbb{F}_3[Y]/\phi_\alpha$ is isomorphic to a direct sum of $\varphi(\alpha)/r$ copies of $\mathbb{F}_{3^r}$. The extraneous factors in r (namely 883, 9041, 327251 and 39551747) arise from the auxiliary prime q_0. Let

$$m := N/\alpha = q_7 q_8 q_{10} q_{11} \cdots q_{6035} \approx 5.439125 \times 10^{99863};$$

then since $m \mid q_1 \cdots q_e \mid 3^r - 1$, each copy of $\mathbb{F}_{3^r}$ contains a primitive m-th root of unity, so $\mathbb{F}_3[Y]/\phi_\alpha$ contains a principal m-th root of unity. Thus it is possible to multiply in the ring $\mathbb{F}_3[Y, Z]/(\phi_\alpha(Y), Z^m - 1)$ by using DFTs over $\mathbb{F}_3[Y]/\phi_\alpha$.

Remark 3.11. In Example 3.10, every $\ell \in \mathcal{L}$ divides $\operatorname{ord}_{q_i} p$ for some i. It seems likely that this always occurs (at least for large n), but we do not know how to prove this. If it fails for some ℓ, then r may turn out not to be divisible by ℓ, but the proof of Proposition 3.8 shows that we still have $q_h \mid p^r - 1$ for every $h = 1, \dots, e$.

4. Faster polynomial multiplication

The goal of this section is to prove Theorem 1.1. We will describe a recursive routine PolynomialMultiply, that takes as input integers $r, t \geqslant 1$, a prime p, and polynomials $U_1, \dots, U_t, V \in \mathbb{F}_p[X]/(X^r - 1)$, and computes the products $U_1 V, \dots, U_t V$. Its running time is denoted by $\mathsf{C}_{\mathrm{poly}}(t, r, p)$. Note that the input polynomials $U_1, \dots, U_t, V$ are expected to be supplied consecutively on the input tape (first U_1, then U_2, and so on), and the outputs $U_1 V, \dots, U_t V$ should also be written consecutively to the output tape.

The role of the parameter t is to allow us to amortise the cost of transforming the fixed operand V across t products. This optimisation (borrowed from [14] and [15]) saves a constant factor in time at each recursion level of the main algorithm. Altogether the algorithm will perform $2t + 1$ transforms: $t + 1$ forward transforms

for $U_1, \ldots, U_t$ and V, followed by t inverse transforms to recover the products $U_1V, \ldots, U_tV$.

To simplify the analysis, it is convenient to introduce the normalisation

$$\mathsf{C}^{\star}_{\mathrm{poly}}(r,p) := \sup_{t \geqslant 1} \frac{\mathsf{C}_{\mathrm{poly}}(t,r,p)}{(2t+1)r \lg p \lg(r \lg p)}.$$

We certainly have $\mathsf{M}_p(n) < \mathsf{C}_{\mathrm{poly}}(1, 2n, p) + O(n \lg p)$, so to prove Theorem 1.1 it is enough to show that

$$\mathsf{C}^{\star}_{\mathrm{poly}}(r,p) = O(4^{\max(0, \log^* r - \log^* p)} K_{\mathbb{Z}}^{\log^* p}). \tag{4.1}$$

The algorithms presented in this section perform many auxiliary multiplications and divisions involving 'small' integers and polynomials. We assume that all auxiliary divisions are reduced to multiplication via Newton's method [23, Ch. 9], so that the cost of a division (by a monic divisor) is at most a constant multiple of the cost of a multiplication of the same bit size. We also assume that, unless otherwise specified, all auxiliary multiplications are handled using the integer and polynomial variants of the Schönhage–Strassen algorithm. The complexity in the integer case is given by

$$\mathsf{M}(n) = O(n \lg n \lg \lg n), \tag{4.2}$$

and in the polynomial case by (1.4).

We first discuss a subroutine Transform that handles DFTs over rings of the form $\mathcal{R}_{p,\alpha} := \mathbb{F}_p[Y]/\phi_\alpha(Y)$, where p is a prime and $\alpha \geqslant 1$. It takes as input p and α, positive integers t and n such that n is odd and relatively prime to α, a principal n-th root of unity $\omega \in \mathcal{R}_{p,\alpha}$, and t input sequences $(a_{s,0}, \ldots, a_{s,n-1}) \in \mathcal{R}^n_{p,\alpha}$ for $s = 1, \ldots, t$. Its output is the sequence of transforms $(\hat{a}_{s,0}, \ldots, \hat{a}_{s,n-1}) \in \mathcal{R}^n_{p,\alpha}$ with respect to ω, for $s = 1, \ldots, t$. Just like PolynomialMultiply, the input and output sequences are stored consecutively on the tape.

Let $\mathsf{T}(t, n, \alpha, p)$ denote the running time of Transform. The following result shows how to reduce the DFT problem to an instance of PolynomialMultiply.

Proposition 4.1. *We have*

$$\mathsf{T}(t,n,\alpha,p) < \mathsf{C}_{\mathrm{poly}}(t, n\alpha, p) + O(tn\alpha \lg \alpha \lg \lg \alpha \lg p \lg \lg p \lg \lg \lg p).$$

Proof. Let $\mathcal{R} := \mathcal{R}_{p,\alpha}$. We use Bluestein's method to reduce each DFT to the problem of computing a certain product $f_s(Z)g(Z)$ in $\mathcal{R}[Z]/(Z^n - 1)$, plus $O(n)$ multiplications in $\mathcal{R}$, where $f_s(Z)$ and $g(Z)$ are defined as in Section 2.2. By (4.2) and (1.4), each multiplication in $\mathcal{R}$ costs

$$O((\alpha \lg \alpha \lg \lg \alpha)(\lg p \lg \lg p \lg \lg \lg p)) \tag{4.3}$$

bit operations. To handle the products $f_s(Z)g(Z)$, we first lift the polynomials from $\mathbb{F}_p[Y,Z]/(\phi_\alpha(Y), Z^n - 1)$ to $\mathbb{F}_p[Y,Z]/(Y^\alpha - 1, Z^n - 1)$ (for example, by zero-padding in Y up to degree α). We then compute their images under the isomorphism

$$\mathbb{F}_p[Y,Z]/(Y^\alpha - 1, Z^n - 1) \cong \mathbb{F}_p[X]/(X^{n\alpha} - 1)$$

provided by Lemma 2.2; this costs altogether $O(tn\alpha \lg \alpha \lg p)$ bit operations. We call PolynomialMultiply to compute the products in $\mathbb{F}_p[X]/(X^{n\alpha} - 1)$, at a cost of $\mathsf{C}_{\mathrm{poly}}(t, n\alpha, p)$ bit operations. We evaluate the inverse of the above isomorphism to bring the products back to $\mathbb{F}_p[Y,Z]/(Y^\alpha - 1, Z^n - 1)$. Finally, we reduce modulo $\phi_\alpha(Y)$ to obtain the desired products in $\mathcal{R}[Z]/(Z^n - 1)$; the cost of each of these divisions is given by (4.3). □

We now return to multiplication in $\mathbb{F}_p[X]/(X^r-1)$. Our implementation of PolynomialMultiply chooses one of two algorithms, depending on the size of r relative to p. For $r \leqslant p^2$ it uses the straightforward Kronecker substitution method described in Section 1. By (1.3) this yields the bound

$$\mathsf{C}_{\mathrm{poly}}(t,r,p) = O(t\,\mathsf{M}(r\lg p)) = O(tr\lg p \lg(r\lg p) K_{\mathbb{Z}}^{\log^*(r\lg p)})$$

and hence

$$\mathsf{C}^\star_{\mathrm{poly}}(r,p) = O(K_{\mathbb{Z}}^{\log^*(p^2\lg p)}) = O(K_{\mathbb{Z}}^{\log^* p}), \qquad r \leqslant p^2. \tag{4.4}$$

Therefore (4.1) holds in this case.

For $r > p^2$, most of the work will be delegated to a subroutine AdmissibleMultiply, which is defined as follows. It takes as input an integer $t \geqslant 1$, a prime p, a p-admissible length N, and polynomials $U_1, \ldots, U_t, V \in \mathbb{F}_p[X]/(X^N-1)$, and computes the products $U_1V, \ldots, U_tV$. In other words, it has the same interface as as PolynomialMultiply, but it only works for p-admissible lengths. We denote its running time by $\mathsf{C}_{\mathrm{ad}}(t,N,p)$. As above we also define the normalisation

$$\mathsf{C}^\star_{\mathrm{ad}}(N,p) := \sup_{t\geqslant 1} \frac{\mathsf{C}_{\mathrm{ad}}(t,N,p)}{(2t+1)N\lg p \lg(N\lg p)}.$$

The reduction from PolynomialMultiply to AdmissibleMultiply in the case $r > p^2$ is given by the following proposition.

Proposition 4.2. *There is an absolute constant $z_3 > 0$ with the following property. For any prime p and any integer $r > \max(z_3, p^2)$, there exists a p-admissible length N in the interval*

$$2r < N < \left(1 + \frac{1}{\lg r}\right) 2r \tag{4.5}$$

such that

$$\mathsf{C}^\star_{\mathrm{poly}}(r,p) < \left(2 + \frac{O(1)}{\lg r}\right) \mathsf{C}^\star_{\mathrm{ad}}(N,p) + O(1). \tag{4.6}$$

Proof. Given as input $U_1, \ldots, U_t, V \in \mathbb{F}_p[X]/(X^r-1)$, our goal is to compute the products $U_1V, \ldots, U_tV$. For sufficiently large r we may apply Proposition 3.4 with $n := 2r$ to find a p-admissible length N such that (4.5) holds. Since $N > 2r$, we may simply zero-pad to reduce each problem to multiplication in $\mathbb{F}_p[X]/(X^N-1)$. This yields

$$\mathsf{C}_{\mathrm{poly}}(t,r,p) < \mathsf{C}_{\mathrm{ad}}(t,N,p) + O(tr\lg p) + 2^{O((\lg\lg r)^2)},$$

where the $tr\lg p$ term arises from the reduction modulo X^r-1, and the last term from Proposition 3.4. Dividing by $(2t+1)r\lg p\lg(r\lg p)$ and taking suprema over $t \geqslant 1$, we find that

$$\mathsf{C}^\star_{\mathrm{poly}}(r,p) < \frac{N\lg(N\lg p)}{r\lg(r\lg p)}\, \mathsf{C}^\star_{\mathrm{ad}}(N,p) + O(1).$$

Finally, since $\lg(N\lg p) \leqslant \lg(r\lg p) + 2$ we obtain

$$\frac{N\lg(N\lg p)}{r\lg(r\lg p)} < 2\left(1+\frac{1}{\lg r}\right)\left(1+\frac{2}{\lg(r\lg p)}\right) < 2 + \frac{O(1)}{\lg r}. \qquad \square$$

The motivation for defining admissible lengths is the following result, which shows how to implement AdmissibleMultiply in terms of a large collection of exponentially smaller instances of PolynomialMultiply.

Proposition 4.3. *There is an absolute constant $z_4 > 0$ with the following property. Let p be a prime and let $N > z_4$ be a p-admissible length. Then there exist integers $r_1, \dots, r_d$ in the interval*

$$2^{(\lg\lg N)^6} < r_i < 2^{(\lg\lg N)^7}, \tag{4.7}$$

and weights $\gamma_1, \dots, \gamma_d > 0$ with $\sum_i \gamma_i = 1$, such that

$$\mathsf{C}^\star_{\mathrm{ad}}(N,p) < \left(2 + \frac{O(1)}{\lg\lg N}\right) \sum_{i=1}^{d} \gamma_i \, \mathsf{C}^\star_{\mathrm{poly}}(r_i, p) + O(1). \tag{4.8}$$

Proof. We are given as input a prime p, a p-admissible length $N = q_0 \cdots q_e$ and polynomials $U_1, \dots, U_t, V \in \mathbb{F}_p[X]/(X^N - 1)$. Our goal is to compute the products $U_1 V, \dots, U_t V$. We will describe a series of reductions that converts this problem to a collection of exponentially smaller multiplication problems, plus overhead of $O(tN \lg N \lg p)$ bit operations incurred during the reductions.

Step 1 — reduce to products over cyclotomic coefficient ring. Invoking Proposition 3.8, we compute a p-admissible divisor α of N, the cyclotomic polynomial $\phi_\alpha \in \mathbb{F}_p[Y]$, and a principal $(q_1 \cdots q_e)$-th root of unity $\omega \in \mathbb{F}_p[Y]/\phi_\alpha$. As $p^2 < N$, this requires at most $2^{O((\lg\lg N)^4)} p^{1+o(1)} < N^{1/2+o(1)}$ bit operations.

Set $\psi_\alpha := (Y^\alpha - 1)/\phi_\alpha \in \mathbb{F}_p[Y]$. Since $Y^\alpha - 1$ has no repeated factors in $\mathbb{F}_p[Y]$, we have $(\phi_\alpha, \psi_\alpha) = 1$. Using the Euclidean algorithm, compute polynomials $\chi_1, \chi_2 \in \mathbb{F}_p[Y]$ of degree at most α such that $\chi_1 \phi_\alpha + \chi_2 \psi_\alpha = 1$; this costs at most $(\alpha \lg p)^{O(1)} < N^{o(1)}$ bit operations.

Let $m := N/\alpha$. As m and α are coprime, Lemma 2.2 provides an isomorphism

$$\mathbb{F}_p[X]/(X^N - 1) \cong \mathbb{F}_p[Y,Z]/(Y^\alpha - 1, Z^m - 1)$$

that may be evaluated in either direction in $O(m\alpha \lg\alpha \lg p)$ bit operations. By (3.4) this simplifies to $O(N(\lg\lg N)^4 \lg p) = O(N \lg N \lg p)$ bit operations. Next, since $(\phi_\alpha, \psi_\alpha) = 1$, there is an isomorphism

$$\mathbb{F}_p[Y]/(Y^\alpha - 1) \cong (\mathbb{F}_p[Y]/\phi_\alpha) \oplus (\mathbb{F}_p[Y]/\psi_\alpha).$$

Using the precomputed polynomials χ_1 and χ_2, we may evaluate the above isomorphism in either direction in

$$\begin{aligned} O((\alpha \lg\alpha \lg\lg\alpha)(\lg p \lg\lg p \lg\lg\lg p)) &= O(\alpha (\lg\lg N)^5 (\lg\lg\lg N)^2 \lg p) \\ &= O(\alpha \lg N \lg p) \end{aligned}$$

bit operations (here we have again used (3.4) and the fact that $p^2 < N$). This isomorphism induces another isomorphism

$$\mathbb{F}_p[Y]/(Y^\alpha - 1, Z^m - 1) \cong (\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m - 1) \oplus (\mathbb{F}_p[Y]/\psi_\alpha)[Z]/(Z^m - 1)$$

by acting on the coefficient of each Z^i separately; it may be evaluated in either direction in $O(m\alpha \lg N \lg p) = O(N \lg N \lg p)$ bit operations. Chaining these isomorphisms together, we obtain an isomorphism

$$\mathbb{F}_p[X]/(X^N - 1) \cong (\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m - 1) \oplus (\mathbb{F}_p[Y]/\psi_\alpha)[Z]/(Z^m - 1)$$

that may be evaluated in either direction in $O(N \lg N \lg p)$ bit operations.

We now use the following algorithm. First, at a cost of $O(tN \lg N \lg p)$ bit operations, apply the above isomorphism to $U_1, \ldots, U_t$ and V to obtain polynomials

$$U_1', \ldots, U_t', V' \in (\mathbb{F}_p[Y]/\phi_\alpha)[Z]/(Z^m - 1),$$
$$\tilde{U}_1', \ldots, \tilde{U}_t', \tilde{V}' \in (\mathbb{F}_p[Y]/\psi_\alpha)[Z]/(Z^m - 1).$$

Second, compute the products $\tilde{U}_1'\tilde{V}', \ldots, \tilde{U}_t'\tilde{V}'$: since $\deg \psi_\alpha < \alpha/\lg N$ by (3.5), each of these products may be converted, via Kronecker substitution, to a product of univariate polynomials in $\mathbb{F}_p[X]$ of degree $O(m\alpha/\lg N) = O(N/\lg N)$ (i.e., map Y to X and Z to $X^{2\deg \psi_\alpha}$). The cost of these multiplications is

$$O(t((N/\lg N)\lg N \lg\lg N)(\lg p \lg\lg p \lg\lg\lg p)) = O(tN \lg N \lg p)$$

bit operations. Third, compute the products $U_1'V', \ldots, U_t'V'$, using the method explained in Step 2 below. Finally, at a cost of $O(tN \lg N \lg p)$ bit operations, apply the inverse isomorphism to the pairs $(U_s'V', \tilde{U}_s'\tilde{V}')$ to obtain the desired products $U_1V, \ldots, U_tV$.

Step 2 — convert to multidimensional convolutions. Let $\mathcal{R} := \mathbb{F}_p[Y]/\phi_\alpha$. In this step our goal is to compute the products $U_1'V', \ldots, U_t'V'$, where $U_1', \ldots, U_t', V' \in \mathcal{R}[Z]/(Z^m - 1)$. We do this by converting each problem to a multidimensional convolution of size $m_d \times \cdots \times m_1$, for a suitable decomposition $m = m_1 \cdots m_d$. For the subsequent complexity analysis, it is important that the m_i are chosen to be somewhat larger than the coefficient size. To achieve this we proceed as follows.

Let $m = \ell_1 \cdots \ell_u$ be the prime factorisation of m. The ℓ_j form a subset of $\{q_1, \ldots, q_e\}$, so by (3.1) we have

$$(\lg N)^3 < \ell_j < 2^{(\lg\lg N)^2} \tag{4.9}$$

for each j. Let $w := \lfloor \frac{2}{5}(\lg\lg N)^5 \rfloor$. We certainly have $u > w$ for large enough N, as (4.9) and (3.4) imply that

$$u > \frac{\log_2 m}{(\lg\lg N)^2} = \frac{\log_2 N - \log_2 \alpha}{(\lg\lg N)^2} > \frac{\log_2 N - (\lg\lg N)^4}{(\lg\lg N)^2} \gg (\lg\lg N)^5.$$

Therefore we may take

$$\begin{aligned} m_1 &:= \ell_1 \cdots \ell_w, \\ m_2 &:= \ell_{w+1} \cdots \ell_{2w}, \\ &\cdots \\ m_{d-1} &:= \ell_{(d-2)w+1} \cdots \ell_{(d-1)w}, \\ m_d &:= \ell_{(d-1)w+1} \cdots \ell_{dw} \ell_{dw+1} \cdots \ell_u, \end{aligned}$$

where $d := \lfloor u/w \rfloor \geqslant 1$. Each m_i is a product of exactly w primes, except possibly m_d, which is a product of at least w and at most $2w - 1$ primes. For large N we have

$$m_i < (2^{(\lg\lg N)^2})^{2w} \leqslant 2^{\frac{4}{5}(\lg\lg N)^7} \tag{4.10}$$

and

$$m_i > ((\lg N)^3)^w \geqslant (2^{\lg\lg N - 1})^{3w} > 2^{(\lg\lg N)^6} \tag{4.11}$$

for all i, and hence

$$d \leqslant \frac{\log_2 m}{(\lg\lg N)^6} \leqslant \frac{\lg N}{(\lg\lg N)^6}. \tag{4.12}$$

Computing the decomposition $m = m_1 \cdots m_d$ requires no more than $(\lg N)^{O(1)}$ bit operations.

As the m_i are pairwise relatively prime, Corollary 2.3 furnishes an isomorphism

$$\mathcal{R}[Z]/(Z^m - 1) \cong \mathcal{R}[Z_1, \ldots, Z_d]/(Z_1^{m_1} - 1, \ldots, Z_d^{m_d} - 1)$$

that may be computed in either direction in $O((m \lg m)(\alpha \lg p)) = O(N \lg N \lg p)$ bit operations. Therefore we may use the following algorithm. First, at a cost of $O(tN \lg N \lg p)$ bit operations, compute the images

$$U_1'', \ldots, U_t'', V'' \in \mathcal{R}[Z_1, \ldots, Z_d]/(Z_1^{m_1} - 1, \ldots, Z_d^{m_d} - 1)$$

of $U_1', \ldots, U_t', V'$ under the above isomorphism. Next, as explained in Step 3 below, compute the products $U_1''V'', \ldots, U_t''V''$. Finally, apply the inverse isomorphism to recover the products $U_1'V', \ldots, U_t'V'$; again this costs $O(tN \lg N \lg p)$ bit operations.

Step 3 — reduce to DFTs over $\mathcal{R}$. In this step our goal is to compute the products $U_1''V'', \ldots, U_t''V''$, where $U_1'', \ldots, U_t''$ and V'' are as above. Let $\omega_i := \omega^{q_1 \cdots q_e / m_i}$ for $i = 1, \ldots, d$, where ω is the principal $(q_1 \cdots q_e)$-th root of unity in $\mathcal{R}$ computed in Step 1. According to the discussion in Section 2.2, the desired multidimensional convolutions may be computed by performing $t+1$ multidimensional m-point DFTs with respect to the evaluation points $(\omega_1^{j_1}, \ldots, \omega_d^{j_d})$, followed by tm pointwise multiplications in $\mathcal{R}$, and then t multidimensional m-point inverse DFTs and tm divisions by m. The total cost of the pointwise multiplications and divisions is

$$O(tm(\alpha \lg \alpha \lg \lg \alpha)(\lg p \lg \lg p \lg \lg \lg p)) = O(tN \lg N \lg p)$$

bit operations.

Each of the $2t + 1$ multidimensional DFTs may be converted to a collection of one-dimensional DFTs of lengths $m_1, \ldots, m_d$ by the method explained in Section 2.2. Note that the inputs must be rearranged so that the data to transform along each dimension may be accessed sequentially. Let $1 \leqslant i \leqslant d$, and consider the transforms of length m_i. Treating each input vector as a sequence of $m_{i+1} \cdots m_d$ arrays of size $m_i \times (m_1 \cdots m_{i-1})$, we must transpose each array into an array of size $(m_1 \cdots m_{i-1}) \times m_i$, perform m/m_i DFTs of length m_i, and then transpose back to the original ordering. The total cost of all these transpositions is

$$O(tm\alpha \lg p \textstyle\sum_i \lg m_i) = O(tN \lg p \lg m) = O(tN \lg N \lg p)$$

bit operations.

The one-dimensional DFTs over $\mathcal{R}$ are handled by the TRANSFORM subroutine. Combining the contributions from Steps 1, 2 and 3 shows that

$$\mathsf{C}_{\mathrm{ad}}(t, N, p) < (2t + 1) \sum_{i=1}^{d} \mathsf{T}\left(\frac{m}{m_i}, m_i, \alpha, p\right) + O(tN \lg N \lg p).$$

This concludes the description of the algorithm; it remains to establish the overall complexity claim. First, Proposition 4.1 yields

$$\sum_{i=1}^{d} \mathsf{T}\left(\frac{m}{m_i}, m_i, \alpha, p\right) < \sum_{i=1}^{d} \mathsf{C}_{\mathrm{poly}}\left(\frac{m}{m_i}, m_i\alpha, p\right) + O(dm\alpha \lg \alpha \lg \lg \alpha \lg p \lg \lg p \lg \lg \lg p).$$

By (4.12), the last term lies in

$$O(dN(\lg \lg N)^5 (\lg \lg \lg N)^2 \lg p) = O(N \lg N \lg p).$$

Setting $r_i := m_i\alpha$ for $i = 1, \ldots, d$, we obtain

$$\mathsf{C}_{\mathrm{ad}}(t, N, p) < (2t+1) \sum_{i=1}^{d} \mathsf{C}_{\mathrm{poly}}\left(\frac{N}{r_i}, r_i, p\right) + O(tN \lg N \lg p).$$

Notice that (4.7) follows immediately from (4.10), (4.11) and (3.4) (for large N). For the normalised quantities, we have

$$\begin{aligned}\mathsf{C}^{\star}_{\mathrm{ad}}(N, p) &< \sum_{i=1}^{d} \frac{\mathsf{C}_{\mathrm{poly}}\left(\frac{N}{r_i}, r_i, p\right)}{N \lg p \lg(N \lg p)} + O(1) \\ &< \sum_{i=1}^{d} \Big(\frac{2N}{r_i} + 1\Big) \frac{r_i \lg(r_i \lg p)}{N \lg(N \lg p)} \, \mathsf{C}^{\star}_{\mathrm{poly}}(r_i, p) + O(1).\end{aligned}$$

Now observe that

$$\frac{\lg(r_i \lg p)}{\lg(N \lg p)} < \frac{\log_2 m_i + \log_2 \alpha + \lg\lg p + O(1)}{\log_2 N} < \frac{\log_2 m_i + O((\lg\lg N)^4)}{\log_2 m}.$$

Put $\gamma_i := \log_2 m_i / \log_2 m$, so that $\sum_i \gamma_i = 1$. Then (4.11) implies that

$$\frac{\lg(r_i \lg p)}{\lg(N \lg p)} < \left(1 + \frac{O((\lg\lg N)^4)}{\log_2 m_i}\right) \gamma_i < \left(1 + \frac{O(1)}{\lg\lg N}\right) \gamma_i.$$

Moreover, from (4.7) we certainly have

$$\Big(\frac{2N}{r_i} + 1\Big) \frac{r_i}{N} = 2 + \frac{r_i}{N} < 2 + \frac{O(1)}{\lg\lg N}.$$

The desired bound (4.8) follows immediately. □

Combining Proposition 4.2 and Proposition 4.3, we obtain the following recurrence inequality for $\mathsf{C}^{\star}_{\mathrm{poly}}(r, p)$. (This is identical to Theorem 7.1 of [15], but with the constant 8 replaced by 4.)

Proposition 4.4. *There are absolute constants $z_5, C_2, C_3 > 0$ and a logarithmically slow function $\Phi : (z_5, \infty) \to \mathbb{R}$ with the following property. For any prime p and any integer $r > \max(z_5, p^2)$, there exist positive integers $r_1, \ldots, r_d < \Phi(r)$, and weights $\gamma_1, \ldots, \gamma_d > 0$ with $\sum_i \gamma_i = 1$, such that*

$$\mathsf{C}^{\star}_{\mathrm{poly}}(r, p) < \left(4 + \frac{C_2}{\lg\lg r}\right) \sum_{i=1}^{d} \gamma_i \, \mathsf{C}^{\star}_{\mathrm{poly}}(r_i, p) + C_3. \tag{4.13}$$

Proof. We first apply Proposition 4.2 to construct a p-admissible length N such that (4.5) and (4.6) both hold; then we apply Proposition 4.3 to construct integers $r_1, \ldots, r_d$ and weights $\gamma_1, \ldots, \gamma_d$ satisfying (4.7) and (4.8). Define $\Phi(x) := 2^{(\log\log x)^8}$; then certainly $r_i < 2^{(\lg\lg 3r)^7} < \Phi(r)$ for large r. The bound (4.13) follows immediately by substituting (4.8) into (4.6). □

Now we may prove our main result for multiplication in $\mathbb{F}_p[X]$. The proof is very similar to that of [15, Thm. 1.1].

Proof of Theorem 1.1. We have already noted that $\mathsf{C}^{\star}_{\mathrm{poly}}(r, p) = O(K_{\mathbb{Z}}^{\log^* p})$ in the region $r \leqslant p^2$ (see (4.4)). To handle the case $r > p^2$, let z_5, C_2, C_3 and $\Phi(x)$ be as

in Proposition 4.4. Increasing z_5 if necessary, we may assume that $z_5 > \exp(\exp(1))$ and that $\Phi(x) \leqslant x - 1$ for all $x > z_5$. For each prime p, set $\sigma_p := \max(z_5, p^2)$ and

$$L_p := \max(C_3, \max_{2 \leqslant r \leqslant \sigma_p} \mathsf{C}^\star_{\mathrm{poly}}(r,p)) = O(K_{\mathbb{Z}}^{\log^* p}).$$

Now apply Proposition 2.1 with $K = 4$, $B = C_2/4$, $\mathcal{S} = \{1, 2, \ldots\}$, $\ell = 2$, $x_0 = z_5$, $\sigma = \sigma_p$, $L = L_p$, and $T(r) = \mathsf{C}^\star_{\mathrm{poly}}(r,p)$.

The first part of the recurrence for $T(y)$ is satisfied due to the definition of L_p, and the second part due to Proposition 4.4. We conclude that $\mathsf{C}^\star_{\mathrm{poly}}(r,p) = O(L_p\, 4^{\log^* r - \log^* \sigma_p})$ for $r > p^2$. Since $\log^* \sigma_p = \log^* p + O(1)$ and $L_p = O(K_{\mathbb{Z}}^{\log^* p})$, we obtain the desired bound $\mathsf{C}^\star_{\mathrm{poly}}(r,p) = O(4^{\log^* r - \log^* p} K_{\mathbb{Z}}^{\log^* p})$ for $r > p^2$. □

Acknowledgments

The authors thank Grégoire Lecerf for his comments on a draft of this paper. The first author was supported by the Australian Research Council (DP150101689 and FT160100219).

References

1. L. M. Adleman, C. Pomerance, and R. S. Rumely, *On distinguishing prime numbers from composite numbers*, Ann. of Math. (2) **117** (1983), no. 1, 173–206. MR 683806 (84e:10008)
2. R. Agarwal and J. Cooley, *New algorithms for digital convolution*, IEEE Transactions on Acoustics, Speech, and Signal Processing **25** (1977), no. 5, 392–410.
3. R. C. Baker, G. Harman, and J. Pintz, *The difference between consecutive primes. II*, Proc. London Math. Soc. (3) **83** (2001), no. 3, 532–562. MR 1851081
4. L. I. Bluestein, *A linear filtering approach to the computation of discrete Fourier transform*, IEEE Transactions on Audio and Electroacoustics **18** (1970), no. 4, 451–455.
5. A. Bostan, P. Gaudry, and É. Schost, *Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator*, SIAM J. Comput. **36** (2007), no. 6, 1777–1806. MR 2299425 (2008a:11156)
6. P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic complexity theory*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 315, Springer-Verlag, Berlin, 1997, With the collaboration of Thomas Lickteig. MR 1440179 (99c:68002)
7. D. G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Inform. **28** (1991), no. 7, 693–701. MR 1129288 (92i:68068)
8. M. Fürer, *Faster integer multiplication*, STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing, ACM, New York, 2007, pp. 57–66. MR 2402428 (2009e:68124)
9. ———, *Faster integer multiplication*, SIAM J. Comput. **39** (2009), no. 3, 979–1005. MR 2538847 (2011b:68296)
10. I. J. Good, *The interaction algorithm and practical Fourier analysis*, J. Roy. Statist. Soc. Ser. B **20** (1958), 361–372. MR 0102888 (21 #1674)
11. D. Harvey and J. van der Hoeven, *Faster integer multiplication using short lattice vectors*, Proceedings of the Thirteenth Algorithmic Number Theory Symposium, Open Book Series 2 (R. Scheidler and J. Sorenson, eds.), Mathematical Sciences Publishers, Berkeley, 2019, pp. 293–310.
12. D. Harvey and J. van der Hoeven, *Faster integer and polynomial multiplication using cyclotomic coefficient rings*, `https://arxiv.org/abs/1712.03693v1`, 2017.
13. D. Harvey, J. van der Hoeven, and G. Lecerf, *Faster polynomial multiplication over finite fields*, technical report, `http://arxiv.org/abs/1407.3361`, 2014.
14. ———, *Even faster integer multiplication*, J. Complexity **36** (2016), 1–30. MR 3530637
15. ———, *Faster polynomial multiplication over finite fields*, J. ACM **63** (2017), no. 6, 52:1–52:23.

16. C. H. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, Reading, MA, 1994. MR 1251285 (95f:68082)
17. K. Prachar, *Über die Anzahl der Teiler einer natürlichen Zahl, welche die Form $p-1$ haben*, Monatsh. Math. **59** (1955), 91–97. MR 0068569
18. A. Schönhage, *Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2*, Acta Informat. **7** (1976/77), no. 4, 395–398. MR 0436663
19. A. Schönhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 0292344 (45 #1431)
20. V. Shoup, *On the deterministic complexity of factoring polynomials over finite fields*, Inform. Process. Lett. **33** (1990), no. 5, 261–267. MR 1049276 (91f:11088)
21. ———, *Searching for primitive roots in finite fields*, Math. Comp. **58** (1992), no. 197, 369–380. MR 1106981
22. L. H. Thomas, *Using computers to solve problems in physics*, Applications of digital computers **458** (1963), 42–57.
23. J. von zur Gathen and J. Gerhard, *Modern computer algebra*, third ed., Cambridge University Press, Cambridge, 2013. MR 3087522

E-mail address: d.harvey@unsw.edu.au

School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia

E-mail address: vdhoeven@lix.polytechnique.fr

Laboratoire d'informatique, UMR 7161 CNRS, École polytechnique, 91128 Palaiseau Cedex, France