Fast Multivariate Multipoint Evaluation Over All Finite Fields

Vishwas Bhargava* Sumanta Ghosh† Zeyu Guo‡ Mrinal Kumar§
Chris Umans¶

Abstract

Multivariate multipoint evaluation is the problem of evaluating a multivariate polynomial, given as a coefficient vector, simultaneously at multiple evaluation points. In this work, we show that there exists a deterministic algorithm for multivariate multipoint evaluation over any finite field $\mathbb{F}$ that outputs the evaluations of an m-variate polynomial of degree less than d in each variable at N points in time

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log |\mathbb{F}|)$$

for all $m \in \mathbb{N}$ and all sufficiently large $d \in \mathbb{N}$.

A previous work of Kedlaya and Umans (FOCS 2008, SICOMP 2011)

achieved the same time complexity when the number of variables m is at most $d^{o(1)}$ and had left the problem of removing this condition as an open problem. A recent work of Bhargava, Ghosh, Kumar and Mohapatra (STOC 2022) answered this question when the underlying field is not *too* large and has characteristic less than $d^{o(1)}$. In this work, we remove this constraint on the number of variables over all finite fields, thereby answering the question of Kedlaya and Umans over all finite fields.

Our algorithm relies on a non-trivial combination of ideas from three seemingly different previously known algorithms for multivariate multipoint evaluation, namely the algorithms of Kedlaya and Umans, that of Björklund, Kaski and Williams (IPEC 2017, Algorithmica 2019), and that of Bhargava, Ghosh, Kumar and Mohapatra, together with a result of Bombieri and Vinogradov from analytic number theory about the distribution of primes in an arithmetic progression.

We also present a second algorithm for multivariate multipoint evaluation that is completely elementary and in particular, avoids the use of the Bombieri–Vinogradov Theorem. However, it requires a mild assumption that the field size is bounded by an exponential-tower in d of bounded *height*. More specifically, our second algorithm solves the multivariate multipoint evaluation problem over a finite field $\mathbb{F}$ in time

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log |\mathbb{F}|)$$

for all $m \in \mathbb{N}$ and all sufficiently large $d \in \mathbb{N}$, provided that the size of the finite field $\mathbb{F}$ is at most $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of this tower of exponentials is fixed.

*Department of Computer Science, Rutgers University, Piscataway, NJ 08854. Research supported in part by the Simons Collaboration on Algorithms and Geometry and NSF grant CCF-1909683. Email: vishwas1384@gmail.com

†Department of Computing and Mathematical Sciences, Caltech. Email: besusumanta@gmail.com

‡Department of Computer Science, UT Austin. Email: zguotcs@gmail.com

§Department of Computer Science & Engineering, IIT Bombay. Email: mrinal@cse.iitb.ac.in

¶Department of Computing and Mathematical Sciences, Caltech. Email: umans@cs.caltech.edu

Contents

1 Introduction

We study the problem of multivariate multipoint evaluation: given an m-variate polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ of degree less than d in each variable, and N points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$, output $f(\mathbf{a}_1), f(\mathbf{a}_2), \ldots, f(\mathbf{a}_N)$. Here $\mathbb{F}$ is the underlying field. The input polynomial $f(\mathbf{x})$ is given by its coefficient vector. Therefore, the overall input can be represented by a list of $(d^m + mN)$ elements in $\mathbb{F}$. A trivial algorithm for this problem is to evaluate $f(\mathbf{x})$ at each $\mathbf{a}_i$ separately. Since evaluating $f(\mathbf{x})$ at each $\mathbf{a}_i$ takes $d^m \cdot \mathsf{poly}(d, m)$ operations over $\mathbb{F}$, this algorithm needs $Nd^m \cdot \mathsf{poly}(d, m)$ $\mathbb{F}$-operations in total. For $N = \Theta(d^m)$, the time complexity of this algorithm is quadratic with respect to the input size. Therefore, a natural algorithmic question here is to seek faster algorithms for this problem. Of particular interest would be to have an algorithm for this problem whose time complexity is *nearly linear*, more specifically $(d^m+N)^{1+o(1)}$ (multiplied by lower-order $\mathsf{poly}(d, n, \log|\mathbb{F}|)$ terms), with respect to the input size.

In addition to its innate appeal as a fundamental and natural question in computational algebra, fast algorithms for multivariate multipoint evaluation are closely related to fast algorithms for other important algebraic problems such as polynomial factorization and modular composition. For a detailed discussion on these connections, we refer to the work of Kedlaya and Umans [KU11]. In a recent work, Bhargava, Ghosh, Kumar and Mohapatra [BGKM21] used the special structure of their algorithm for multivariate multipoint evaluation to show an upper bound on the rigidity of Vandermonde matrices and very efficient algebraic data structures for the polynomial evaluation problem over finite fields.

For the setting of univariate polynomials, Borodin and Moenck [BM74] showed that the multipoint evaluation can be solved in nearly linear time. Their algorithm is short, simple and elementary, and proceeds via an application of the Fast Fourier Transform (FFT). However, this approach does not seem to extend when the number of variables exceeds one; in fact, even when the number of variables is two. However, for the multivariate case, when the input points form a product set, one can naturally extend the ideas in Borodin and Moenck [BM74] to get a nearly linear time algorithm for this problem. But, when the input points are arbitrary, getting a sub-quadratic algorithm for multipoint evaluation seems to be significantly more difficult. In fact, about three decades after Borodin and Moenck's work, Nüsken and Ziegler [NZ04] proved that multipoint evaluation can be solved in most $O(d^{\omega_2/2+1})$ operations for $m = 2$ and $N = d^2$, where ω_2 is the exponent for multiplying a $d \times d$ and a $d \times d^2$ matrix. The work [NZ04] extends to general m and gives an algorithm for multipoint evaluation that performs $O(d^{\omega_2/2\cdot(m-1)+1})$ field operations.

Two significant milestones in this line of work are the results of Umans [Uma08] and Kedlaya and Umans [KU11]. Umans [Uma08] gave a nearly linear time (that is, $(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log|F|)$-time) *algebraic* algorithm[1] for this problem over finite fields, provided that the characteristic of the field and the number of variables are at most $d^{o(1)}$. Later, Kedlaya and Umans [KU11] gave a nearly linear time *non-algebraic* algorithm for *all* finite fields, but they also need $m = d^{o(1)}$. In a recent work, Bhargava, Ghosh, Kumar and Mohapatra [BGKM21] improve the result of Umans [Uma08] by removing the restriction on m over finite fields whose characteristics are small and sizes are not too large. More specifically, they gave a nearly linear time algebraic algorithm for multivariate multipoint evaluation, provided that the characteristic of the field is $d^{o(1)}$ and the size of the field is at most $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of this tower of exponentials is fixed. Another closely related result is a recent work of Björklund, Kaski and Williams [BKW19] who (among other results) gave an algorithm for multivariate multipoint evaluation, but their time complexity depends polynomially on the field size (and not polynomially on the logarithm of the field size), and instead of d^m, their time complexity is nearly linear in D^m where D is the total degree of the polynomial. Nevertheless,

[1] Algorithms for multivariate multipoint evaluation can be divided into two categories: (1) *algebraic algorithms*, where we are only allowed to perform arithmetic operations over the underlying field $\mathbb{F}$, and (2) *non-algebraic algorithms*, where we are allowed to do bit operations. The algorithms of Kedlaya and Umans [KU11] and those in this paper are not algebraic, whereas the algorithm of Borodin and Moenck [BM74], that of Umans [Uma08], and that of Bhargava, Ghosh, Kumar and Mohapatra [BGKM21] are all algebraic.

their results play a crucial role in proving the results of this paper and we will discuss them in more detail in Section 2.2.

Thus, from the context of previous work, a very natural and interesting open question is to design an algorithm, for the problem of multivariate multipoint evaluation that runs in nearly linear time and works for all finite fields and all ranges of the number of variables. Indeed, Kedlaya and Umans [KU11] mention this as an open problem.

In this work, we answer this question by giving two different algorithms for multivariate multipoint evaluation over finite fields. While our first algorithm works over all finite fields, the second algorithm still requires that the field size is not too large in terms of d. We now state our results and discuss the pros and the cons of the two algorithms and compare them to the algorithms known in prior work. Both our algorithms happen to be non-algebraic, i.e. we need more than just arithmetic operations over the underlying field.

We now state our results and discuss how they compare with prior work.

1.1 Our Results

We state our main result as follows.

Theorem 1.1. *There is a deterministic algorithm that given the coefficient vector of an m-variate polynomial $f(\mathbf{x})$ of degree less than d in each variable over a finite field $\mathbb{F}$ and N points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$, outputs $f(\mathbf{a}_1), f(\mathbf{a}_2), \ldots, f(\mathbf{a}_N)$ in time*

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log |\mathbb{F}|),$$

for all $m \in \mathbb{N}$ and all sufficiently large $d \in \mathbb{N}$.

Remark. *Throughout this paper, when we say d is* sufficiently large*, it means $d = \omega(1)$.* ◊

The proof of the above theorem crucially relies on a deep result from analytic number theory, known as the Bombieri–Vinogradov Theorem [Bom65, Vin65], related to the distribution of primes in arithmetic progressions.

We also give a different algorithm for multivariate multipoint evaluation that avoids the Bombieri–Vinogradov Theorem and is completely elementary, but it requires the size of the finite field to be not too large: at most $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of this tower of exponentials is fixed. In other words, it removes the restriction on the characteristic of the field in the work of Bhargava *et.al.* [BGKM21], but via a non-algebraic algorithm.

We remark that neither of our algorithms is algebraic, and in particular, we crucially rely on working with the bit representation of the inputs. To obtain an algebraic algorithm for multivariate multipoint evaluation, for large m, and over all finite fields is a fundamental algebraic problem that continues to remain open.

Below, in Table 1, we compare the results in this paper with the previously known results.

Table 1: Comparison with Prior Results

Multivariate Multipoint Evaluation over a Finite Field $\mathbb{F}_q$ of Characteristic p				
Results	Time	Algorithm Type	Field Constraint	Variable
[Uma08]	$(d^m+N)^{1+o(1)}\cdot$ $\mathsf{poly}(m,d,\log q)$	algebraic	$p \leq d^{o(1)}$	$m \leq d^{o(1)}$
[KU11]	$(d^m+N)^{1+o(1)}\cdot$ $\mathsf{poly}(m,d,\log q)$	non-algebraic	all finite fields	$m \leq d^{o(1)}$
[BGKM21]	$(d^m+N)^{1+o(1)}\cdot$ $\mathsf{poly}(m,d,\log q)$	algebraic	$p \leq d^{o(1)}$, $q \leq (\exp(\exp(\cdots(\exp(d)))))$, where the height of this tower of exponentials is fixed	any m
This work (Algorithm 2)	$(d^m+N)^{1+o(1)}\cdot$ $\mathsf{poly}(m,d,\log q)$	non-algebraic	all finite fields	any m
This work (Algorithm 5)	$(d^m+N)^{1+o(1)}\cdot$ $\mathsf{poly}(m,d,\log q)$	non-algebraic	$q \leq (\exp(\exp(\cdots(\exp(d)))))$, where the height of this tower of exponentials is fixed	any m

2 An Overview of the Proofs

In this section, we give an overview of the main ideas in our algorithms. At a high level, our algorithms rely on ideas from three of the recent prior works on multivariate multipoint evaluation, namely that of Kedlaya and Umans [KU11], that of Björklund, Kaski and Williams [BKW19], and a recent work of Bhargava, Ghosh, Kumar and Mohapatra [BGKM21]. We start by giving a brief outline of these.

We start with some necessary notation. Let $\mathbb{F}$ be a finite field and let $f \in \mathbb{F}[\mathbf{x}]$ be an m-variate polynomial of degree less than d in each variable, and let $\{\mathbf{a}_i : i \in [N]\}$ be a set of N inputs in $\mathbb{F}^m$. Our goal is to evaluate f on each $\mathbf{a}_i$. For simplicity, we focus on the case when the underlying field $\mathbb{F}$ is a prime field, i.e. $\mathbb{F} = \mathbb{F}_p$ for some prime p. The case of extension fields is handled in a very similar manner, with a few technicalities.

A starting observation is that multivariate multipoint evaluation has a nearly linear time algorithm (over all fields) when the set of evaluation points forms a product set (see Lemma 3.10), and more generally when the set of evaluation points is *close* to a product set. At a high level, each of the algorithms in [KU11, BKW19, BGKM21] proceeds via a very efficient reduction from multivariate multipoint evaluation over an arbitrary set of points to multivariate multipoint evaluation over product sets. However, despite this common high level structure, the details of the reductions involved are fairly different in each of the three algorithms, thereby giving these algorithms their features, both desirable and undesirable. We now elaborate a bit more on these reductions.

2.1 The Algorithm of Kedlaya and Umans

To solve the problem efficiently over a finite field $\mathbb{F}$, Kedlaya and Umans [KU11] first reduce an instance of the multivariate multipoint evaluation problem over $\mathbb{F}$ to an instance of the same problem over a ring of the form $\mathbb{Z}/r\mathbb{Z}$. Then they use their efficient algorithm for multivariate multipoint evaluation problem over $\mathbb{Z}/r\mathbb{Z}$ to solve it. Finally, from the evaluations over $\mathbb{Z}/r\mathbb{Z}$, they recover the original evaluations over $\mathbb{F}$. So, we first describe their algorithm over ring $\mathbb{Z}/r\mathbb{Z}$.

The algorithm over $\mathbb{Z}/r\mathbb{Z}$: In their algorithm, Kedlaya and Umans [KU11, §4.2] start by *lifting* their problem instance over $\mathbb{Z}/r\mathbb{Z}$ to an instance over integers. They do this by just viewing $\mathbb{Z}/r\mathbb{Z}$ as the set of integers $\{0, 1, \ldots, r-1\}$ and this naturally maps a polynomial $f(\mathbf{x})$

over $\mathbb{Z}/r\mathbb{Z}$ to a polynomial $F(\mathbf{x})$ with coefficients in $\mathbb{Z}$. Similarly, this also gives a natural map from an input point $\mathbf{a} \in (\mathbb{Z}/r\mathbb{Z})^m$ to a point $\tilde{\mathbf{a}} \in \mathbb{Z}^m$. Clearly, for every $\mathbf{a} \in (\mathbb{Z}/r\mathbb{Z})^m$ and polynomial f, $f(\mathbf{a}) = F(\tilde{\mathbf{a}}) \mod r$. Thus, it suffices to solve this lifted instance over integers. Yet another property of this lifted instance is that the integer $F(\tilde{\mathbf{a}})$ is a non-negative integer of magnitude less than $M = d^m(r-1)^{dm}$ since each coefficient of F and each coordinate of $\tilde{\mathbf{a}}$ are in $\{0, 1, \ldots, r-1\}$, and the total degree of F is less than or equal to $(d-1)m$. Thus, to compute $F(\tilde{\mathbf{a}})$, it suffices to compute $F(\tilde{\mathbf{a}}) \mod M$. Kedlaya and Umans now proceed by finding distinct small primes $p_1, p_2, \ldots, p_k$ such that $\prod_{i\in[k]} p_i > M$, evaluating the polynomial $f_j(\mathbf{x}) = F \mod p_j$ at the point $\mathbf{b}_j = \tilde{\mathbf{a}} \mod p_j$ and [2] then combining the values $f_1(\mathbf{b}_1), f_2(\mathbf{b}_2), \ldots, f_k(\mathbf{b}_k)$ using the Chinese Remainder Theorem. The correctness follows from the observation that for every $j \in [k]$, $f_j(\mathbf{b}_j) = F(\tilde{\mathbf{a}}) \mod p_j$. The advantage of this *multimodular* reduction is that if the primes p_j are very small (for instance, if all these primes are close to d), then the set of evaluation points of interest, that were initially *scattered* sparsely in $\mathbb{F}_p^n$ are now mapped to points that are packed densely in the space $\mathbb{F}_{p_j}^m$, which is a product set. Thus, we can use the simple multidimensional FFT to evaluate f_j on all of $\mathbb{F}_{p_j}^m$ for every j, and then combine the outcome using the Chinese Remainder Theorem. For $m < d^{o(1)}$, this indeed gives a nearly linear time algorithm for multivariate multipoint evaluation. This constraint on the number of variables m is due to a term of the form $(dm)^m$ in the final running time of the algorithm which is nearly linear in the input size only if m is small. This $(dm)^m$ essentially appears because the product of primes $p_1, p_2, \ldots, p_k$ chosen in this reduction must exceed M, and hence, the largest of these primes p_k must be $\Omega(\log M) = \Omega(dm \log r)$, and thus evaluating a polynomial f_k on all of $\mathbb{F}_{p_k}^m$ requires at least $p_k^m = \Omega(d^m m^m)$ time. Recursive application of this process leads to smaller primes but the improved dependence is on the $\log r$ factor and this $(dm)^m$ factor continues to persist in the eventual bound on the running time. Thus, one approach towards a faster algorithm for multipoint evaluation over $\mathbb{Z}/r\mathbb{Z}$ would be to replace this step of evaluating f_j on all of $\mathbb{F}_{p_j}^m$ in [KU11] with a faster subroutine, in particular, something that runs in nearly linear time in the input size even for large m.

Our first algorithm in this paper does precisely this. In order to obtain this gain, it crucially relies on ideas in an algorithm of Björklund, Kaski and Williams [BKW19] which we discuss in Section 2.2 and a very careful choice of primes to do Chinese Remaindering with, in the multimodular reduction discussed above. Together, these steps lead to an improvement in running time and give us an algorithm that runs in nearly linear time even when the number of variables is large.

For our second algorithm, we introduce a slightly different modification in the framework of Kedlaya and Umans. Instead of working modulo small primes as in [KU11], which as discussed above, forces us to pick primes as large as dm, we work modulo powers of distinct primes in the multimodular reduction step. Thus, it seems conceivable that we can now work with much smaller primes than in the original algorithm, since instead of having the condition that the product of these primes is larger than M as in [KU11], we now need that the product of powers of these primes is larger than M. However, we still need efficient algorithms for multivariate multipoint evaluation over rings of the form $\mathbb{Z}/p^k\mathbb{Z}$ for small primes p and large $k \in \mathbb{N}$. To handle this subproblem, we extend the derivative-based techniques used in the algorithm of Bhargava et al. [BGKM21] for fields of small characteristic so that they work over rings of the form $\mathbb{Z}/p^k\mathbb{Z}$ for small primes p and large $k \in \mathbb{N}$.

The advantage of this strategy over our first algorithm is that this gives us a completely elementary algorithm, and the disadvantage is that for this algorithm to run in nearly linear time, as desirable, the underlying ring $\mathbb{Z}/r\mathbb{Z}$ needs to be somewhat small. This issue also affects the original algorithm of Bhargava et al. [BGKM21] and seems somewhat inherent to this style of an argument.

[2] In other words, f_j is obtained from F by reducing each of its coefficients modulo p_j and $\mathbf{b}_j$ is obtained by reducing each of the coordinates of $\tilde{\mathbf{a}}$ modulo p_j.

The algorithm over all finite fields: We now give an outline of the algorithm of Kedlaya and Umans for extensions of prime fields.

Let $\mathbb{F}$ be the underlying finite field such that $|\mathbb{F}| = p^e$ for some prime p and positive integer e. Then, we can assume that $\mathbb{F}$ is represented by $\mathbb{F}_p[z]/(E(z))$ for some degree e irreducible monic polynomial $E(z)$ over $\mathbb{F}_p$. Let $f(\mathbf{x})$ be the input polynomial over $\mathbb{F}$ with m variables and degree less than d in each variable and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ be the input points. Observe that each coefficient of $f(\mathbf{x})$ and each coordinate of $\mathbf{a}_i$'s are polynomial in $\mathbb{F}_p[z]$ of degree at most $e-1$. Like the previous case, Kedlaya and Umans [KU11] lift $f(\mathbf{x})$ to a polynomial $F(\mathbf{x}) \in \mathbb{Z}[z][\mathbf{x}]$ and $\mathbf{a}_i$ to $\tilde{\mathbf{a}}_i \in \mathbb{Z}[z]^m$ by naturally identifying each element of $\mathbb{F}$ to a polynomial in $\mathbb{Z}[z]$ of degree at most $e-1$ and coefficients are in the set of integers $\{0, 1, \ldots, p-1\}$. This reduces the problem of computing $f(\mathbf{a}_i)$ for all $i \in [N]$ to the problem of computing $F(\tilde{\mathbf{a}}_i)$ for all $i \in [N]$ since from $f(\mathbf{a}_i)$ is $F(\tilde{\mathbf{a}}_i)$ modulo p and $E(z)$.

Let $M = d^m(e(p-1))^{(d-1)m+1} + 1$. One can observe that the coefficients of $F(\tilde{\mathbf{a}}_i)$, viewed as a polynomial in z are all less than M. It follows from the fact that the evaluation of $F(\tilde{\mathbf{a}}_i)$ at $z = 1$ is at most $M-1$. Therefore, we can recover $F(\tilde{\mathbf{a}}_i)$ by finding the M-base representation of the evaluation of $F(\tilde{\mathbf{a}}_i)$ at $z = M$. Also, note that the degree of $F(\tilde{\mathbf{a}}_i)$ in z is at most $(e-1)dm$, hence the evaluation of $F(\tilde{\mathbf{a}}_i)$ at $z = M$ is less than $r = M^{(e-1)dm+1}$. Thus, computing $F(\tilde{\mathbf{a}}_i)$ modulo $z - M$ and r is sufficient for computing $F(\tilde{\mathbf{a}}_i)$. This implies that they need to solve the following instance of the multivariate multipoint evaluation problem over the ring $\mathbb{Z}/r\mathbb{Z}$: the input polynomial is $F(\mathbf{x})$ modulo r and $z - M$, the evaluation points are $\tilde{\mathbf{a}}_i$ modulo r and $z - M$. Now they invoke their multivariate multipoint evaluation over the ring $\mathbb{Z}/r\mathbb{Z}$ and get $F(\tilde{\mathbf{a}}_i)$ at $z = M$ for all $i \in [N]$.

2.2 The Algorithm of Björklund, Kaski and Williams

In a nutshell, the algorithm of Björklund et al. [BKW19] proceeds via constructing a set $K \subseteq \mathbb{F}_p^m$ such that

- The size of K is not too large and K is (*close* to) a product set.
- For every $\mathbf{a} \in \mathbb{F}_p^m$, there is a curve $C_{\mathbf{a}}$ of low degree (in fact, a low degree univariate polynomial map) that passes through the point $\mathbf{a}$ and intersects the set K on at least p points [3].

These sets K can be thought of as a natural higher degree analog of Kakeya sets over finite fields from discrete geometry. Indeed, Björklund et al. refer to the set K as high degree Kakeya sets, where the degree of the set is defined to be the maximum over the degrees of the curves $C_{\mathbf{a}}$ over all $\mathbf{a} \in \mathbb{F}_p^m$.

Given such a Kakeya set K, Björklund et al.proceed by evaluating f on all points in K fast, using the multidimensional FFT algorithm. This is the preprocessing phase of the algorithm. Then, for an arbitrary point $\mathbf{a} \in \mathbb{F}_p^m$, they compute $f(\mathbf{a})$ by considering the univariate polynomial $R(y)$ obtained by taking the restriction f on the curve $C_{\mathbf{a}}$. From the properties of the set K, we know the curve $C_{\mathbf{a}}$ intersects the set K on at least p points. Thus, if the degree of $R \leq \deg(f) \cdot \deg(C_{\mathbf{a}})$ is less than p, then we can recover the polynomial R from the evaluations of f on K computed in the preprocessing step and using univariate polynomial interpolation. The quantitative bounds for this approach are therefore crucially determined by the size of the set K and the degree of the curve $C_{\mathbf{a}}$.

Björklund et al. showed that for every $u \in \mathbb{N}$ such that $u+1$ divides $p-1$, there is a Kakeya set K of degree u of size at most $((p-1)/(u+1)+1)^{m+1}$. This divisibility condition ensures the existence of a multiplicative subgroup of $\mathbb{F}_p^*$ of size $(p-1)/(u+1)$ and set K is based on this subgroup. Thus, if $\tilde{d}$ denotes $(p-1)/(u+1)$, then we can evaluate the polynomial f on the K in time $\tilde{d}^m$, which is nearly linear in the input size if $\tilde{d} \leq d^{1+o(1)}$. However, note that in this case, u is around $p/\tilde{d}$, and hence, the degree of the restriction R of f on a curve of degree u has total

[3] This notion of a curve passing through a point here is slightly different to that in other related works like [BGKM21]. However, for the sake of simplicity, we gloss over this technical detail right now.

degree $udm = pm \cdot \frac{d}{\tilde{d}}$. Thus, if $pm \cdot \frac{d}{\tilde{d}} > p$, we cannot hope to recover R from its evaluations on just p points. To address this issue, we combine the above strategy in [BKW19] with an idea in [BGKM21] where instead of evaluating just f on K, we evaluate all its (Hasse) derivatives of order at most $m \cdot \frac{d}{\tilde{d}}$ on K in the preprocessing phase. There are at most $\binom{m+m\cdot\frac{d}{\tilde{d}}}{m}$ such derivatives and this leads to an additional multiplicative factor of $\binom{m+m\cdot\frac{d}{\tilde{d}}}{m}$ in the final running time, but if $\tilde{d}$ is not too small compared to d, for instance, $\tilde{d} = \Theta(d)$, this binomial coefficient is at most $\exp(O(m))$ which is $d^{o(m)}$ for all growing d. Thus, with this stronger guarantee in the preprocessing step, we are guaranteed to have higher multiplicity information available to us in the local computation step. So, we can now hope to uniquely recover a univariate polynomial of degree higher than p from this information (via Hermite interpolation). However, since the degree of the univariates we have here is larger than p, this Hermite interpolation step runs in time polynomially bounded in the underlying field size p and not just polynomially bounded in $\log p$ as would have been desirable.

To summarise, if there exists an $u \in \mathbb{N}$ such that $(p-1)/(u+1) = \tilde{d}$, where $\tilde{d}$ is close to d, e.g. $\tilde{d} = \Theta(d)$, then we have an algorithm for evaluating m-variate polynomials of degree less than d in each variable on any N points in $\mathbb{F}_p^m$ in time $\mathsf{poly}(p, d, m) \cdot (d^m + N)^{1+o(1)}$. Thus, this is nearly linear time, when the field size p is not too large.

Having discussed these prior results, we are now ready to give an outline of our algorithms. We start with the first algorithm.

2.3 The First Algorithm

As discussed earlier in this section, the plan for our algorithm is to somehow replace the multi-dimensional FFT step in the algorithm of Kedlaya and Umans [KU11] (over rings of the form $\mathbb{Z}/r\mathbb{Z}$) with the Kakeya-set-based algorithm above over a field $\mathbb{F}_{p_j}$. However, in order to effectively use the Kakeya-set-based algorithm outlined in the previous section to obtain nearly linear time algorithms for multipoint evaluation, we need to ensure two properties.

- The underlying field size p_j is small. For instance, we would need $p_j = (d^m + N)^{o(1)}$ for a nearly linear time algorithm.
- There is a natural number u such that $u+1$ divides $p_j - 1$ and $(p_j - 1)/(u+1)$ is an integer close to d.

In fact, instead of the second condition here, it suffices if there is a small $t \in \mathbb{N}$ such that there exists a $u \in \mathbb{N}$ such that $u+1$ divides $p_j^t - 1$ and $(p_j^t - 1)/(r+1) = d^{1+o(1)}$, since we can always view the problem over $\mathbb{F}_p$ as a problem over an extension of $\mathbb{F}_p$. However, we need the degree of the extension to be small in order to get useful final quantitative bounds.

The first condition about the primes p_j being small does not appear too difficult to ensure in isolation and in particular, is also true for the algorithm of Kedlaya and Umans. However, the second divisibility condition seems trickier to guarantee even with the flexibility of working over low degree extensions of $\mathbb{F}_{p_j}$ as outlined earlier in this section. In particular, it is not clear to us if for every pair d, p_j, there always exists small t such that $p_j^u - 1$ has a divisor in the vicinity of d.

Getting around these technical difficulties is the main technical content of our algorithm. In a nutshell, we proceed by following the multimodular reduction step of Kedlaya and Umans, but via a careful choice of primes $p_1, p_2, \ldots, p_k$ (as opposed to picking a sufficiently large number of small primes as in [KU11]). This careful choice preserves the fact that these primes are all small (at most $\mathsf{poly}(d, m, \log p)$) and additionally guarantees that the divisibility condition needed to invoke the Kakeya-set-based framework of [BKW19]. More formally, we choose $p_1, p_2, \ldots, p_k$ so that they are all at most $\mathsf{poly}(d, m, \log p)$, their product exceeds $M = d^m(p-1)^{dm}$ and there exists a $\tilde{d} \in [0.8d, d]$ such that for every $j \in [k]$, $\tilde{d}$ divides $p_j - 1$. Thus, we can use the Kakeya-set-based framework outlined in Section 2.2, with the parameter u_j to be set equal to $(p_j - 1)/\tilde{d} - 1$. This satisfies both the conditions highlighted earlier, and the final running time of this algorithm does indeed turn out to be nearly linear in the input size. The details can be

found in Section 5. Once we have this algorithm for multipoint evaluation over the rings of the form $\mathbb{Z}/r\mathbb{Z}$, we use exactly the same strategy as Kedlaya and Umans did to solve this problem over all finite fields. For details see Section 6.

Thus, if we can find distinct primes $p_1, p_2, \ldots, p_k$ with the properties outlined above, we would be done. However, it is not immediately clear how to do find such a set of numbers efficiently, or whether such a collection of primes and the parameter $\tilde{d}$ should even exist. The appearance of the parameter $\tilde{d} = \Theta(d)$ is also slightly mysterious. For instance, it would be aesthetically nice if $\tilde{d}$ would have been equal to d. Perhaps surprisingly, we do not know how to even show the existence of primes $p_1, p_2, \ldots, p_k$ satisfying the desired properties with $\tilde{d} = d$! We now outline our approach to finding such primes and the parameter $\tilde{d}$. However, for a start, let us attempt to do this with $\tilde{d} = d$ and try to understand the issues that arise.

The intuition on showing the existence of such primes follows from the observation that if d divides $p_j - 1$ for each $j \in [k]$ then, each of the primes $p_1, p_2, \ldots, p_k$ lies in the arithmetic progression (AP) $A_d = (1, 1 + d, 1 + 2d, \ldots)$. It follows from a classical theorem of Dirichlet (see Chapter 5 in [Ked15] for more details) that this arithmetic progression A_d indeed contains an infinite number of primes for every $d \in \mathbb{N}$. Thus, if we take k to be sufficiently large, then there exist primes $p_1, p_2, \ldots, p_k$ each congruent to 1 modulo d such that their product is greater than $M = d^m(p-1)^{dm}$. However, it is not enough for our application. We also need to show that these primes are not too large, e.g. each $p_i \leq \mathsf{poly}(d, m, \log p)$, and that they can be found efficiently. For this, it would be sufficient to show that not only does the arithmetic progression A_d contains an infinite number of primes, but the set of primes in A_d is also a sufficiently dense subset of A_d. The prime number theorem gives such a statement for the progression A_1, i.e. for the set of natural numbers and here, a similar statement for arbitrary arithmetic progressions is needed. An unconditional bound on the density of primes in an arithmetic progression A_d is given by the well-known Siegel-Walfisz theorem [Sie35, Wal36] which implies a lower bound on the number of primes less than x in the AP A_d for all $x \geq 0$ with $x > 2^{d^\epsilon}$ for any constant ϵ. However, this estimate does not appear to be sufficient for us, since for the algorithm, we need the magnitude of these primes to be at most $\mathsf{poly}(d, m, \log p)$ and not exponentially growing in d, and it is not clear if such a guarantee can be obtained directly from this theorem. An improved lower bound on the density of primes in arithmetic progressions is known under the Generalized Riemann Hypothesis, and this would have been sufficient for our applications, except for the fact that the result would be conditional. For the unconditional result in this paper, we rely on the following theorem of Bombieri and Vinogradov, which gives an improved lower bound on the density of primes in an AP on average. For $x > 0, t \in \mathbb{N}$, let $\pi(x, t)$ be the number of primes less than x in the AP starting at 1 and with common difference t, $\pi(x)$ denote the number of primes less than x, and $\phi : \mathbb{N} \to \mathbb{N}$ be the Euler Totient function. Various versions of this theorem can be found in literature, for instance, [Bom65, Vin65], Theorem 18.1 in [Ked15]. Here we rely on the bound in equation 1.1. in [May20].

Theorem 2.1 (Bombieri–Vinogradov)**.** *For any fixed $a > 0$, there exist constants $c = c(a)$ and $b = b(a)$ such that for all sufficiently large $x > 0$,*

$$\sum_{t \leq d} \left| \pi(x, t) - \frac{\pi(x)}{\phi(t)} \right| \leq cx(\log x)^{-a},$$

where $d \leq x^{1/2}(\log x)^{-b}$.

Thus, if x is sufficiently large compared to d, e.g. $x = d^3$, this theorem can be viewed as saying that on average (over $t \in \mathbb{N}, t \leq d$), an AP with common difference t contains at least $\frac{\pi(x)}{\phi(t)} - cxd^{-1}(\log x)^{-a}$ primes less than x. Clearly, $\phi(t) \leq t \leq d$ and $\pi(x) = \Theta(x/\log x)$ by the prime number theorem. Thus, if we take $a > 1$, the number of primes less than x is at least $\Omega(\pi(x)/d)$. For our final argument, we combine this average-case statement about the density of primes in an AP with a standard application of Markov's inequality to deduce that there exists a $\tilde{d} \in [0.8d, d]$ such that the AP with common difference $\tilde{d}$ has at least $\Omega(\pi(x)/\tilde{d})$ many primes less than x. By choosing x to be a sufficiently large polynomial in $d, m, \log p$, we get

precisely what we want: sufficiently many primes $p_1, p_2, \ldots, p_k$, each at most $\mathsf{poly}(d, m, \log p)$ in absolute value such that their product exceeds M and they are all congruent to 1 modulo $\tilde{d}$, for $\tilde{d} = \Theta(d)$. This application of Markov's inequality is precisely why we have to settle for working with the quantity $\tilde{d}$ and not d itself.

2.4 The Second Algorithm

In this section, we give a brief overview of our second algorithm. It implies that Theorem 1.1 holds as long as the size of the finite field is bounded by $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of this tower of exponentials is fixed via an elementary algorithm. In particular, this algorithm does not rely on the Bombieri-Vinogradov theorem necessary for the first algorithm.

For simplicity, we first explain our algorithm over rings of the form $\mathbb{Z}/r\mathbb{Z}$, or $\mathbb{Z}/r^s\mathbb{Z}$ for some $s \leq m$. This covers the case of prime finite fields $\mathbb{F}_p$ by choosing $r = p$ and $s = 1$. After that, we briefly explain how to extend the algorithm to make it work over non-prime finite fields and certain extension rings of $\mathbb{Z}/r\mathbb{Z}$.

The algorithm over $\mathbb{Z}/r\mathbb{Z}$: Recall that Kedlaya and Umans [KU11, §4.2] use multimodular reduction together with the Chinese Remainder Theorem to reduce the multivariate multipoint evaluation problem over $\mathbb{Z}/r\mathbb{Z}$ to that over $\mathbb{F}_{p_j}$ for a collection of small primes p_j. As discussed in Section 2.1, for the Chinese Remainder Theorem, the primes p_j need to be chosen such that $\prod_{i\in[k]} p_i > M := d^m(r-1)^{dm}$. The problem here is that, as the primes p_j are distinct, the largest prime would have order $O(\log M) = O(dm \log r)$. The $\log r$ factor can be further reduced by repeating the multimodular reduction. However, the dm factor persists. As a consequence, the time complexity of the Kedlaya–Umans algorithm has a factor $(dm)^m$, which is nearly linear in d^m only when $m = d^{o(1)}$.

In our algorithm, we introduce the new idea of using the *prime powers* p_j^m as the moduli for Chinese remaindering instead of the primes p_j. That is, we compute the evaluations over the rings $\mathbb{Z}/p_j^m\mathbb{Z}$ and then combine them via Chinese Remainder Theorem to obtain the evaluations over the integers. Assuming this can be done, then we only need to choose the primes p_j such that $\prod_{i\in[k]} p_i^m > M$. So the largest prime may have order $O(\frac{1}{m}\log M) = O(d \log r)$, which is independent of m.

Now, to make this idea work, we need a fast algorithm for multivariate multipoint evaluation over $\mathbb{Z}/p_j^m\mathbb{Z}$, for small primes p_j. In particular, if we have an algorithm over $\mathbb{Z}/p_j^m\mathbb{Z}$ that runs in time $(p_j^m+N)^{1+o(1)}$, then, overall, we have an algorithm that runs in time $(d^m(\log r)^m+N)^{1+o(1)}$. Note that this has already enabled us to get rid of the m^m factor in the running time as in [KU11]. So, up to the factor of $(\log r)^m$ in the running time, we seem to have made some progress and we soon elaborate further on how to reduce this $(\log r)^m$ factor further. But first, we note that naively evaluating the polynomial at all points in $(\mathbb{Z}/p_j^m\mathbb{Z})^m$ would be extremely inefficient, as the size of $(\mathbb{Z}/p_j^m\mathbb{Z})^m$ is exponential in m^2. So, we need a significantly faster algorithm for multivariate multipoint evaluation over $\mathbb{Z}/p_j^m\mathbb{Z}$ to have any hope of making this strategy work.

In their algorithm, Kedlaya and Umans [KU11] deal with the $(\log r)^m$ factor by recursively applying the multimodular reduction a few times. So, to reduce the $(\log r)^m$ in the discussion above, we could also try to do something similar. We already see that one application of the reduction reduces the modulus r to p_j^m for a collection of primes p_j, where $\prod_{i\in[k]} p_i > d(r-1)^d$. Fix a prime p_j and suppose we want to apply the multimodular reduction again. We may lift the instance over $\mathbb{Z}/p_j^m\mathbb{Z}$ to an instance over the integers, and then reduce it modulo $p_i'^m$ for a collection of primes p_i'. The problem here is that, if we simply lift the evaluation points from $(\mathbb{Z}/p_j^m\mathbb{Z})^m$ to $\{0, 1, \ldots, p_j^m - 1\}^m$, we would have an upper bound $M' = d^m(p_j^m-1)^{dm}$ for the evaluations over the integers, which is too large for us. The primes p_i' would have to satisfy $\prod_i p_i' > M'^{1/m} = d(p_j^m-1)^d$, and then the order of the largest prime must depend (at least polynomially) on m.

We address the above two challenges, namely that of obtaining a fast multipoint evaluation algorithm over $\mathbb{Z}/p_j^m\mathbb{Z}$ that does not require evaluating on all of $\mathbb{Z}/p_j^m\mathbb{Z}^m$ and that of reducing

the factor $(\log r)^m$ using the following observation: over $\mathbb{Z}/r^s\mathbb{Z}$, the evaluation of an m-variate polynomial $f(\mathbf{x})$ at a point $\mathbf{a} \in (\mathbb{Z}/r^s\mathbb{Z})^m$ can be derived from the evaluations of the Hasse derivatives of $f(\mathbf{x})$ of sufficiently high order at another point $\mathbf{b} \in (\mathbb{Z}/r^s\mathbb{Z})^m$, provided that the coordinates of $\mathbf{a} - \mathbf{b}$ are all multiples of r. Intuitively, this means if $\mathbf{a}$ and $\mathbf{b}$ are "close enough", then we can learn the evaluation of $f(\mathbf{x})$ at $\mathbf{a}$ from the evaluations at $\mathbf{b}$ of all the Hasse derivatives of f of sufficiently high order.

Formally, for all $\mathbf{e} \in \mathbb{N}^m$, let $\overline{\partial}_{\mathbf{e}}(f) \in (\mathbb{Z}/r^s\mathbb{Z})[\mathbf{x}]$ be the Hasse derivative of $f(\mathbf{x})$ with respect to the monomial $\mathbf{x}^{\mathbf{e}}$. For $\mathbf{a}, \mathbf{b} \in (\mathbb{Z}/r^s\mathbb{Z})^m$, we get from Taylor's expansion of $f(\mathbf{x})$ at $\mathbf{b}$ that

$$f(\mathbf{a}) = \sum_{\mathbf{e} \in \mathbb{N}^m} \overline{\partial}_{\mathbf{e}}(f)(\mathbf{b})(\mathbf{a} - \mathbf{b})^{\mathbf{e}}.$$

Suppose the coordinates of $\mathbf{a} - \mathbf{b}$ are all multiples of r. In this case, observe that $(\mathbf{a} - \mathbf{b})^{\mathbf{e}} = 0$ in $\mathbb{Z}/r^s\mathbb{Z}$ for all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 \geq s$. Hence,

$$f(\mathbf{a}) = \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < s} \overline{\partial}_{\mathbf{e}}(f)(\mathbf{b})(\mathbf{a} - \mathbf{b})^{\mathbf{e}}. \tag{2.2}$$

So we may compute $f(\mathbf{a})$ from the evaluations of Hasse derivatives $(\overline{\partial}_{\mathbf{e}}(f)(\mathbf{b}))_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < s}$.

We apply this idea to resolve the above two issues. First, in a base case of the recursive algorithm, instead of evaluating $f(\mathbf{x})$ at all points in $(\mathbb{Z}/p_j^m\mathbb{Z})^m$, we evaluate the Hasse derivatives $\overline{\partial}_{\mathbf{e}}(f)$ at the points in S^m using a fast evaluation algorithm for product sets, where S is the subset of $\mathbb{Z}/p_j^m\mathbb{Z}$ represented by $\{0, 1, \ldots, p_j - 1\}$. Note that for any $\mathbf{a} \in (\mathbb{Z}/p_j^m\mathbb{Z})^m$, we may find $\mathbf{b} \in S^m$ such that the coordinates of $\mathbf{a} - \mathbf{b}$ are multiples of p_j. Then $f(\mathbf{a})$ can be computed from $\overline{\partial}_{\mathbf{e}}(f)(\mathbf{b})$ using (2.2). The advantage of this is that the size of S^m is only p_j^m, which is much smaller than the size $p_j^{m^2}$ of the whole set $(\mathbb{Z}/p_j^m\mathbb{Z})^m$.

Similarly, when applying the multimodular reduction over a ring $\mathbb{Z}/p_j^m\mathbb{Z}$, the idea above allows us to use a small yet non-exact lift of each evaluation point $\mathbf{a}_i$. Namely, suppose $\tilde{\mathbf{a}}_i \in \mathbb{Z}^m$ is the unique lift of $\mathbf{a}_i \in (\mathbb{Z}/p_j^m\mathbb{Z})^m$ with coordinates in $\{0, 1, \ldots, p_j^m - 1\}$. We compute $\tilde{\mathbf{a}}'_i \in \{0, 1, \ldots, p_j - 1\}^m$ whose coordinates are obtained by reducing the corresponding coordinates of $\tilde{\mathbf{a}}_i$ modulo p_j. Then $\tilde{\mathbf{a}}'_i$ is a lift of some $\mathbf{a}'_i \in (\mathbb{Z}/p_j^m\mathbb{Z})^m$ such that the coordinates of $\mathbf{a}_i - \mathbf{a}'_i$ are all multiples of p_j. We compute the evaluation $\overline{\partial}_{\mathbf{e}}(f)(\mathbf{a}'_i)$ at the point $\mathbf{a}'_i$ (instead of $\mathbf{a}$), and then $f(\mathbf{a}_i)$ can be computed from $\overline{\partial}_{\mathbf{e}}(f)(\mathbf{a}'_i)$ using (2.2). The advantage of evaluating at $\mathbf{a}'_i$ instead of $\mathbf{a}_i$ is that the coordinates of its lift $\tilde{\mathbf{a}}'_i$ are bounded by $p_j - 1$ instead of $p_j^m - 1$. This translates into a better bound for the primes that we choose in multimodular reduction, thereby resolving the second issue.

Finally, at each level of the recursive algorithm, we need to evaluate not only $f(\mathbf{x})$, but also the Hasse derivatives $\overline{\partial}_{\mathbf{e}}(f)$ of order less than m. In addition, we need to solve the subproblem for each prime p_j. This means the number of subproblems blows up by a factor of $2^{O(m)} \cdot O(d \log r)$ each time. However, as we assume the original r (= the field size when r is prime) is reasonably bounded in terms of d, it takes only a constant number of rounds to reduce r to $d^{1+o(1)}$. So the total blow-up is reasonably controlled, and we obtain a nearly linear time algorithm when d is sufficiently large. For details, see Section 7.

Comparison with the first algorithm: Compared to our first algorithm, which uses the ideas of generalized Kakeya sets and the Bombieri–Vinogradov theorem, our second algorithm uses a different idea, namely the Chinese Remainder Theorem with prime powers as the moduli. At a high level, this may be seen as an analogue of the "method of multiplicities" applied to the ring $\mathbb{Z}$ and polynomial rings over $\mathbb{Z}$. To see this, note that for a univariate polynomial $f(x)$ over a field, knowing the evaluations of all (Hasse) derivatives $f^{(i)}(x)$ of order $< s$ at a point a is equivalent to knowing the remainder of f modulo the power $(x - a)^s$. So from an ideal-theoretic point of view, the idea of applying the Chinese Remainder Theorem to learn an integer from its remainders modulo prime powers is analogous to applying Hermite interpolation to learn a univariate polynomial from the evaluations of its Hasse derivatives, the latter playing a crucial role in [BGKM21].

The algorithm over finite fields (and extension rings of $\mathbb{Z}/r\mathbb{Z}$): With further ideas, we extend our algorithm so that it works over arbitrary finite fields. In fact, our algorithm works more generally over a ring $(\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$, where $r \geq 2$ is an integer and $E(z) \in (\mathbb{Z}/r\mathbb{Z})[z]$ is a monic irreducible polynomial of degree $e \geq 1$.

Kedlaya and Umans [KU11, §4.3] described a reduction that reduces the problem of multivariate multipoint evaluation over $(\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ to that over $\mathbb{Z}/r'\mathbb{Z}$ for some integer r'. Unfortunately, the integer r' there is too large for us, being exponential in m^2. This is not a bottleneck in [KU11], as their algorithm over $\mathbb{Z}/r\mathbb{Z}$ already has a factor m^m in its time complexity. However, it is a problem for us, so we cannot directly use the reduction in [KU11].

To achieve our claimed time complexity, we design a more efficient reduction, which reduces the evaluation problem to that over $\mathbb{Z}/r'^m\mathbb{Z}$, where r' is independent of m. The basic idea is lifting the problem instance to an instance over $\mathbb{Z}[z]$, and then reducing it modulo r'^m and $(z-j)^m$ for a small number of integers j. Here the idea of raising $z-j$ and r' to their m-th powers helps us keep r' small, and in particular, independent of m. See Section 8 for the details of the algorithm and a more thorough overview.

3 Preliminaries

Define $\mathbb{N} = \{0, 1, \dots\}$, $\mathbb{N}^+ = \{1, 2, \dots\}$, $[n] = \{1, 2, \dots, n\}$, and $[\![n]\!] = \{0, 1, \dots, n-1\}$. The cardinality of a set S is denoted by $|S|$.

All rings in this paper are commutative rings with unity. For univariate polynomials $f(x), g(x)$ over a ring R such that $g(x)$ is monic of positive degree, there exist unique $h(x), r(x) \in R[x]$ such that $f(x) = g(x)h(x) + r(x)$ and $\deg(r) < \deg(g)$ [Lan02, §IV.1, Theorem 1.1]. Define $f(x) \bmod g(x) := r(x)$, which can be computed using polynomially many R-operations via long division.

By $\mathbf{x}$ and $\mathbf{z}$, we denote the variable tuples $(x_1, \dots, x_m)$ and $(z_1, \dots, z_m)$, respectively. For any $\mathbf{e} = (e_1, \dots, e_m) \in \mathbb{N}^m$, $\mathbf{x}^{\mathbf{e}}$ denotes the monomial $\prod_{i=1}^m x_i^{e_i}$. By $|\mathbf{e}|_1$, we denote the sum $e_1 + \cdots + e_m$.

For every positive integer k, $k!$ denotes $\prod_{i=1}^k i$. For $k = 0$, $k!$ is defined as 1. For two non-negative integers i and k with $k \geq i$, $\binom{k}{i}$ denotes $\frac{k!}{i!(k-i)!}$. For $k < i$, $\binom{k}{i} = 0$. For $\mathbf{a} = (a_1, \dots, a_m), \mathbf{b} = (b_1, \dots, b_m) \in \mathbb{N}^m$, $\binom{\mathbf{a}}{\mathbf{b}} = \prod_{i=1}^m \binom{a_i}{b_i}$.

Proposition 3.1. *For any two positive integers i and k with $k \geq i$,*

$$\binom{k}{i} \leq \left(\frac{ke}{i}\right)^i.$$

For a proof, see [Juk11, Chapter 1]. All logarithms in this paper are with respect to base 2. For a non-negative integer c, $\log^{\circ c}(n)$ denotes the c-times composition of the logarithm function with itself. For example, $\log^{\circ 2}(n) = \log\log(n)$. We denote by $\log^\star(n)$ the smallest non-negative integer c such that $\log^{\circ c}(n) \leq 1$.

We need the following number-theoretic result.

Lemma 3.2 ([KU11, Lemma 2.4])**.** *For all integers $N \geq 2$, the product of the primes $p \leq 16 \log N$ is greater than N.*

3.1 Chinese Remainder Theorem

For our algorithms, we crucially use the Chinese Remainder Theorem. For completeness, we formally state the version we use and refer to Chapter 10 of [vzGG13] for a proof.

Theorem 3.3 (Chinese Remainder Theorem)**.** *Let $n_1, n_2, \dots, n_t$ be pairwise relatively prime natural numbers greater than or equal to 2 and let $u_1, u_2, \dots, u_t$ be arbitrary natural numbers such that for every $i \in [t]$, $u_i \leq n_i - 1$. Then, there is a unique $v \in \mathbb{N}$ with $v < \prod_{i=1}^t n_i$ such that for every $i \in [t]$, $v \equiv u_i \pmod{n_i}$.*

Moreover, there is a deterministic algorithm, that when given $n_1, n_2, \ldots, n_t$ and $u_1, u_2, \ldots, u_t$ as input, outputs v in time at most $\mathsf{poly}(\sum_{i\in[t]} \log n_i)$, i.e., in time polynomial in the input size.

3.2 Hasse Derivatives

In this section, we briefly discuss the notion of Hasse derivatives that plays a crucial role in our results.

Definition 3.4 (Hasse derivative)**.** *Let $f(\mathbf{x})$ be an m-variate polynomial over a commutative ring R. Let $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$. Then, the Hasse derivative of f with respect to the monomial $\mathbf{x}^\mathbf{e}$ is the coefficient of $\mathbf{z}^\mathbf{e}$ in the polynomial $f(\mathbf{x}+\mathbf{z}) \in (R[\mathbf{x}])[\mathbf{z}]$.* ◊

Notations. Suppose that $f(\mathbf{x})$ is an m-variate polynomial over a commutative ring R. For $\mathbf{a} \in \mathbb{N}^m$, denote by $\overline{\partial}_\mathbf{a}(f)$ the Hasse derivative of $f(\mathbf{x})$ with respect to the monomial $\mathbf{x}^\mathbf{a}$. For any non-negative integer k, define

$$\overline{\partial}^{\leq k}(f) := \left\{\overline{\partial}_\mathbf{a}(f) \mid \mathbf{a} \in \mathbb{N}^m \text{ s.t. } |\mathbf{a}|_1 \leq k\right\}$$

and

$$\overline{\partial}^{<k}(f) := \{\overline{\partial}_\mathbf{a}(f) \mid \mathbf{a} \in \mathbb{N}^m \text{ s.t. } |\mathbf{a}|_1 < k\}.$$

For a univariate polynomial $h(t)$ over $\mathbb{F}$ and a non-negative integer k, denote by $h^{(k)}(t)$ the Hasse derivative of $h(t)$ with respect to the monomial t^k, that is, $\mathrm{Coeff}_{z^k}(h(t+z))$.

Next, we mention a useful property of Hasse derivatives.

Proposition 3.5. *Let $f(\mathbf{x})$ be an m-variate polynomial over a commutative ring R. Let $\mathbf{a}, \mathbf{e} \in \mathbb{N}^m$. Then,*

$$\overline{\partial}_\mathbf{e}(f) = \sum_{\mathbf{a}\in\mathbb{N}^m} \binom{\mathbf{a}}{\mathbf{e}} \mathrm{Coeff}_{\mathbf{x}^\mathbf{a}}(f)\mathbf{x}^{\mathbf{a}-\mathbf{e}}.$$

For a proof, see, e.g., [For14, Appendix C].

The following lemma states that Hasse derivatives of polynomials can be computed efficiently.

Lemma 3.6. *Let R be either a finite field or a ring of the form $\mathbb{Z}/r\mathbb{Z}$. There exists an algorithm that given an m-variate polynomial $f(\mathbf{x})$ of individual degree less than d over R and $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 \leq dm$, computes $\overline{\partial}_\mathbf{e}(f)$ in time $O(d^m) \cdot \mathsf{poly}(m, d, \log|R|)$.*

Proof. Let $S = [\![d]\!]^m$. For all $\mathbf{a} \in S$, let $c_\mathbf{a}$ denote the coefficient of monomial $\mathbf{x}^\mathbf{a}$ in f. Then, from Proposition 3.5, we know that

$$\overline{\partial}_\mathbf{e}(f) = \sum_{\mathbf{a}\in S} \binom{\mathbf{a}}{\mathbf{e}} c_\mathbf{a}\mathbf{x}^{\mathbf{a}-\mathbf{e}}.$$

Without loss of generality, we can assume that each coordinate of $\mathbf{e}$ is less than d. Otherwise, $\overline{\partial}_\mathbf{e}(f)$ is a zero polynomial. For any $\mathbf{a} \in S$, $\binom{\mathbf{a}}{\mathbf{e}}$ can be computed in time $\mathsf{poly}(m, d, \log|R|)$. Thus, the time needed to compute $\overline{\partial}_\mathbf{e}(f)$ is $O(d^m) \cdot \mathsf{poly}(m, d, \log|R|)$. □

Remark. *For simplicity, we assume in Lemma 3.6 that R is either a finite field or a finite ring of the form $\mathbb{Z}/r\mathbb{Z}$, as this will be sufficient for us. The same assumption is made in Lemma 3.9 and Lemma 3.10, even though the lemmas and their proofs extend to general rings.* ◊

A useful additional ingredient in the proof of Theorem 1.1 is the following lemma. Semantically, this is an explicit form of the chain rule of Hasse derivatives for the restriction of a multivariate polynomial to a curve of low degree.

Lemma 3.7. *Let $f(\mathbf{x})$ be an m-variate degree d polynomial over a field $\mathbb{F}$, $\mathbf{g}(t) = (g_1, \ldots, g_m)$ where $g_i \in \mathbb{F}[t]$, and $h(t) = f(\mathbf{g}(t))$. For all $i \in [m]$, let $g_i(t + Z) = g_i(t) + Z\tilde{g}_i(t, Z)$ for some $\tilde{g}_i \in \mathbb{F}[t, Z]$. Let $\tilde{\mathbf{g}}(t, Z) = (\tilde{g}_1, \ldots, \tilde{g}_m)$, and for all $\mathbf{e} = (e_1, \ldots, e_m) \in \mathbb{N}^m$, $\tilde{\mathbf{g}}_{\mathbf{e}} = \prod_{i=1}^{m} \tilde{g}_i^{e_i}$. For any $\ell \in \mathbb{N}$, let*

$$h_\ell(t, Z) = \sum_{i=0}^{\ell} Z^i \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 = i} \overline{\partial}_{\mathbf{e}}(f)(\mathbf{g}(t)) \cdot \tilde{\mathbf{g}}_{\mathbf{e}}(t, Z).$$

Then, for every $k \in \mathbb{N}$ with $k \leq \ell$, $h^{(k)}(t) = \mathrm{Coeff}_{Z^k}(h_\ell)$.

We now state a lemma that uses the above lemma for fast evaluation of all the Hasse derivatives of $h(t) = f(\mathbf{g}(t))$ over a finite field $\mathbb{F}_q$.

Lemma 3.8. *Let $f(\mathbf{x})$ be an m-variate, individual degree less than d polynomial over a finite field $\mathbb{F}_q$ and $\mathbf{g}(t) = (g_1, g_2, \ldots, g_m)$ where $g_i \in \mathbb{F}_q[t]$ with degree bounded by r. Then, given access to evaluations of $\overline{\partial}^{\leq 2m}(f)$ on $\mathbb{F}_q$, there exists an algorithm that computes the evaluations of all $\leq 2m$ order Hasse derivatives of the polynomial $h(t) = f(\mathbf{g}(t))$ at all points in $\mathbb{F}_q$ in time $\Theta(1)^m \cdot \mathsf{poly}(q, r, d, m)$.*

The proof of the above lemma (and its promised algorithm) follows directly from Algorithm 4 in [BGKM21] and its correctness, thus is skipped here. The only change is that Algorithm 4 looked at $\leq m$-th order Hasse derivatives, and here we are looking at $\leq 2m$-th order Hasse derivatives. It is an easy exercise to see that the analysis of the algorithm in [BGKM21] extends as it is to this case.

3.3 Hermite Interpolation

The following lemma gives a stronger version of univariate polynomial interpolation, known as Hermite interpolation. To interpolate a univariate polynomial of degree d, we need its evaluations at $d + 1$ distinct points. However, for Hermite interpolation, the number of evaluation points can be less than d, provided that evaluations of Hasse derivatives of the polynomial are available up to a certain order.

Lemma 3.9 (Hermite interpolation)**.** *Let R be either a finite field or a ring of the form $\mathbb{Z}/r\mathbb{Z}$. Let $f(x)$ be a univariate polynomial over R and $e_1, \ldots, e_\ell$ be positive integers such that $d := e_1 + \cdots + e_\ell$ is greater than $\deg(f)$. Let $a_1, a_2, \ldots, a_\ell \in R$ such that for distinct $i, j \in [\ell]$, $a_i - a_j$ has multiplicative inverse in R. For all $i \in [\ell]$ and $j \in [\![e_j]\!]$, let $\beta_{ij} = f^{(j)}(a_i)$. Then given (a_i, β_{ij}) for all $i \in [\ell]$ and $j \in [\![e_j]\!]$, $f(x)$ can be computed in time $\mathsf{poly}(d, \log |R|)$. Equivalently, given $(a_i, f(x) \bmod (x - a_i)^{e_i})$ for all $i \in [\ell]$, $f(x)$ can be computed in time $\mathsf{poly}(d, \log |R|)$.*

Proof. Note $f(x) \bmod (x - a_i)^{e_i} = \sum_{j=0}^{e_j - 1} \beta_{ij}(x - a_i)^j$ for $i \in [\ell]$. So (a_i, β_{ij}) and $(a_i, f(x) \bmod (x - a_i)^{e_i})$ can be computed from each other in time $\mathsf{poly}(d, \log |R|)$, and using either of them as the input is equivalent to using the other.

Next, we show that $f(x)$ can be computed in $\mathsf{poly}(d)$ R-operations given $f_i(x) := f(x) \bmod (x - a_i)^{e_i}$ for $i \in [\ell]$. When R is a field, see [vzGG13, §5.6] for a proof. We give a proof that works for general R. For $i \in [\ell]$, compute the following data. First, compute

$$r_i := \prod_{j \in [\ell] \setminus \{i\}} (x - a_j)^{e_j} \bmod (x - a_i) = \prod_{j \in [\ell] \setminus \{i\}} (a_i - a_j)^{e_j},$$

which is a unit in R as each $a_i - a_j$ is a unit. Then compute $h_i(x) \in R[x]$ such that

$$\prod_{j \in [\ell] \setminus \{i\}} (x - a_j)^{e_j} = r_i - h_i(x)(x - a_i).$$

Let $\lambda_i(x) := r_i^{-1} h_i(x)(x - a_i)$. Compute $\delta_i(x) := 1 - \lambda_i(x)^{e_i}$. As $\lambda_i(x)$ is a multiple of $x - a_i$, we have $\delta_i(x) \equiv 1 \pmod{(x - a_i)^{e_i}}$. As $\delta_i(x)$ is a multiple of $1 - \lambda_i(x) = r_i^{-1} \prod_{j \in [\ell] \setminus \{i\}} (x - a_j)^{e_j}$,

we also have $\delta_i(x) \equiv 0 \pmod{(x-a_j)^{e_j}}$ for $j \in [\ell] \setminus \{i\}$. Finally, compute

$$g(x) := \sum_{i=1}^{\ell} \delta_i(x) f_i(x) \mod \prod_{i=1}^{\ell} (x-a_i)^{e_i}.$$

Then $g(x) \equiv f_i(x) \equiv f(x) \pmod{(x-a_i)^{e_i}}$ for $i \in [\ell]$. It remains to prove $g(x) = f(x)$. We know $g(x) - f(x)$ is a multiple of $(x-a_i)^{e_i}$ for $i \in [\ell]$. For distinct $i, j \in [\ell]$, the proof above constructs a multiple of $(x-a_i)^{e_i}$ whose remainder modulo $(x-a_j)^{e_j}$ is one. In particular, $(x-a_i)^{e_i}$ is multiplicatively invertible modulo $(x-a_j)^{e_j}$. So $(g(x)-f(x))/(x-a_i)^{e_i}$ is still a multiple of $(x-a_j)^{e_j}$ for all $j \in [\ell] \setminus \{i\}$. Repeating this argument shows that $g(x) - f(x)$ is a multiple of the degree-d polynomial $\prod_{i=1}^{\ell}(x-a_i)^{e_i}$. As $\deg(g), \deg(f) < d$, we have $g(x) = f(x)$. □

3.4 Fast Multivariate Multipoint Evaluation for Product Sets

The following lemma states that multivariate multipoint evaluation can be solved very efficiently if the set of evaluation points is a product set.

Lemma 3.10. *Let R be either a finite field or a ring of the form $\mathbb{Z}/r\mathbb{Z}$. There exists an algorithm that given an m-variate polynomial $f(\mathbf{x})$ of individual degree less than d over R and a finite subset S of R, outputs the evaluations $f(\mathbf{a})$ for all $\mathbf{a} \in S^m$ in time $O(d^m + |S|^m) \cdot \mathsf{poly}(m, d, \log |R|)$.*

Proof. If $m = 0$, $f \in R$ is just a scalar, and its evaluation at the only point in S^0 is f itself. So just output f.

Now assume $m > 0$. Compute $f_a := f(x_1, \ldots, x_{m-1}, a)$ for $a \in S$, which can be done in time $O(|S|d^m) \cdot \mathsf{poly}(m, d, \log |R|)$.[4] For each $a \in S$, recursively compute the evaluations $f_a(\mathbf{a})$ for all $\mathbf{a} \in S^{m-1}$. Then output $(f_a(\mathbf{a}))_{\mathbf{a} \in S^{m-1}, a \in S} = (f(\mathbf{a}))_{\mathbf{a} \in S^m}$.

Now we give an upper bound $T(m)$ for the time complexity of the above algorithm. We have $T(0) = O(1)$ and $T(m) = |S| \cdot T(m-1) + T'$ where $T' := O(|S|d^m) \cdot \mathsf{poly}(m, d, \log |R|)$.

When $|S| \le d$, we have $T' = O(d^m) \cdot \mathsf{poly}(m, d, \log |R|)$. In this case, solving the recurrence relation using the fact $|S| \le d$ yields $T(m) = O(d^m) \cdot \mathsf{poly}(m, d, \log |R|)$. When $|S| > d$, we have $T' = O(|S|^m) \cdot \mathsf{poly}(m, d, \log |R|)$, and solving the recurrence relation yields $T(m) = O(|S|^m) \cdot \mathsf{poly}(m, d, \log |R|)$.

It follows that $T(m) = O(d^m + |S|^m) \cdot \mathsf{poly}(m, d, \log |R|)$. □

4 The Necessary Building Blocks

In this section, we set up some of the necessary building blocks for our algorithm.

4.1 Primes in an Arithmetic Progression

The first ingredient we need is the existence of *sufficiently many* primes in the arithmetic progression $A_d = \{1, 1+d, 1+2d, \ldots\}$ that are not too large. When d is small, and x tends to infinity, a well-known result of Dirichlet (Theorem 5.5 in [Ked15]) shows that the density of primes less than x in the arithmetic progression A_d tends to $\Theta(\frac{x}{\phi(d) \log x})$, where ϕ is the Euler totient function. However, for our application, we will need x and d to be *close* to each other and hence it becomes important to carefully look at the error term in the prime counting function for the progression A_d.

While we do not know how to show such a statement, we end up working with a weaker statement that turns out to be sufficient for our application. This weaker statement that we use follows (immediately) from a deep result of Bombieri and Vinogradov that we state now.

[4] One can use FFT-based fast univariate multipoint evaluation [vzGG13] over $R[x_1, \ldots, x_{m-1}]$ to compute all f_a in time $d^{m-1} \cdot \tilde{O}(d + |S|) \cdot \mathsf{poly}(m, \log |R|)$, and the eventual time complexity would be $(d^{m-1} + |S|^{m-1}) \cdot \tilde{O}(d + |S|) \cdot \mathsf{poly}(m, \log |R|)$. For us, the time complexity bound in Lemma 3.10 is good enough.

A more general statement can be found in Theorem 18.1 in [Ked15]. But first, we need some notation. For any $x \geq 0$, we denote by $\pi(x)$ the number of primes less than or equal to x. For $x \geq 0$ and $t \in \mathbb{N}$, we also use $\pi(x,t)$ to denote the number of primes less than or equal to x in the arithmetic progression $A_t = \{1, 1+t, 1+2t, \dots, \}$

We are now ready to state the theorem of Bombieri and Vinogradov that we use. Various versions of the theorem can be found in literature, for instance, [Bom65, Vin65], Theorem 18.1 in [Ked15]. Here we rely on the bound in Equation 1.1 in [May20].

Theorem 4.1 (Bombieri-Vinogradov)**.** *For any fixed $a > 0$, there exist constants $c = c(a)$ and $b = b(a)$ such that for all sufficiently large $x > 0$,*

$$\sum_{t \leq Q} \left| \pi(x,t) - \frac{\pi(x)}{\phi(t)} \right| \leq cx(\log x)^{-a} ,$$

where $Q \leq x^{1/2}(\log x)^{-b}$.

Semantically, Theorem 2.1 says that on average (over $t \leq Q$), the quantity $\left|\pi(x,t) - \frac{\pi(x)}{\phi(t)}\right|$ is bounded by $(cx(\log x)^{-a})$. For our application, we would require a similar statement in the worst-case choice of t. This, however, is not known unconditionally when t is large compared to x[5] (which will turn out to be the case here), unless we assume the Generalized Riemann Hypothesis. Thankfully, it turns out that we have some wriggle room, and we can in fact work with the average-case statement above (up to some small loss in the parameters). More formally, we need the following immediate consequence of Theorem 2.1.

Lemma 4.2. *For any fixed $a > 1$, there exist constants $c = c(a)$ and $b = b(a)$ such that for all sufficiently large $x > 0$, $Q \leq x^{1/2}(\log x)^{-b}$ and $\delta > 1$, there is a $t_0 \in \mathbb{N}$ with $Q(1-2/\delta) \leq t_0 \leq Q$ and*

$$\pi(x, t_0) \geq \frac{x}{4Q \log x} .$$

Proof. From Theorem 2.1, by dividing both sides by Q, we can view the summation as an expectation as t varies uniformly in $[Q] = \{1, 2, \dots, \lfloor Q \rfloor\}$. So, we get

$$\mathop{\mathbb{E}}_{t \in [Q]} \left[\left| \pi(x,t) - \frac{\pi(x)}{\phi(t)} \right| \right] \leq \frac{cx(\log x)^{-a}}{Q} .$$

Now, by Markov's tail bound for the non-negative random variable $\left|\pi(x,t) - \frac{\pi(x)}{\phi(t)}\right|$, we get that for any $\delta > 1$,

$$\Pr_{t \in [Q]} \left[\left| \pi(x,t) - \frac{\pi(x)}{\phi(t)} \right| > \delta \cdot \frac{cx(\log x)^{-a}}{Q} \right] \leq 1/\delta .$$

In particular, there exists an integer $t_0 \in [Q - 2Q/\delta, Q]$ such that

$$\left| \pi(x,t_0) - \frac{\pi(x)}{\phi(t_0)} \right| \leq \delta \cdot \frac{cx(\log x)^{-a}}{Q} .$$

Or, in other words,

$$\pi(x,t_0) \geq \frac{\pi(x)}{\phi(t_0)} - \delta \cdot \frac{cx(\log x)^{-a}}{Q} .$$

Now, since x is sufficiently large, we have that $\pi(x) \geq (1 - o(1))x/\log x \geq x/2 \log x$. So,

$$\pi(x,t_0) \geq \frac{x}{\log x} \left(\frac{1}{2\phi(t_0)} - \delta \cdot \frac{c(\log x)^{1-a}}{Q} \right) .$$

[5] More specifically, we would like x and t to be polynomially related to each other.

Now, since $t_0 \leq Q$, $\phi(t_0) \leq t_0 \leq Q$. So, we have

$$\pi(x, t_0) \geq \frac{x}{Q \log x} \left(\frac{1}{2} - \delta \cdot c(\log x)^{1-a} \right) .$$

Finally, using the fact that x is sufficiently large, and c, δ, a are constants with $a > 1$, we get

$$\pi(x, t_0) \geq \frac{x}{4Q \log x} .$$

□

We now state the following consequence of this lemma that will be directly useful for us in the Chinese Remaindering step of our algorithm.

Lemma 4.3. *Let D, M be natural numbers and let D be sufficiently large. Then, there exists a natural number $\tilde{D} \in [0.8D, D]$ such that there are distinct primes $p_1, p_2, \ldots, p_k$ in the arithmetic progression $A_{\tilde{D}} = (1, 1 + \tilde{D}, 1 + \tilde{D}, \ldots,)$ with the following properties.*

1. *$k \leq D^2 (\log M)^3$*
2. *For every $i \in [k]$, $p_i \leq (D \log M)^3$*
3. *$\prod_{i=1}^{k} p_i > M$*

Moreover, there is a deterministic algorithm that on input D, M outputs $p_1, \ldots, p_k, \tilde{D}$ in time $\mathsf{poly}(D, \log M)$.

Proof. We invoke Lemma 4.2 with the parameters a set to an arbitrary positive constant greater than 1, e.g. $a = 10$, Q set to D, $x = (D \log M)^3$. Note that $D \leq \sqrt{x} (\log x)^{-b}$ for $b = b(a)$ as given by Lemma 4.2 for this choice of parameters. Let the constant δ set to be 10 (any arbitrary constant greater than 2 works). Now, by Lemma 4.2, we get that there is a $\tilde{D} \in [0.8D, D]$ such that

$$\pi(x, \tilde{D}) \geq \frac{x}{4D \log x} .$$

Let us consider the product of k of these primes in the arithmetic progression $\{1, 1 + \tilde{D}, 1 + 2\tilde{D}, \ldots, \}$ for any $k \leq \frac{x}{4D \log x}$. Clearly, each of these primes is at least $\tilde{D}$ (and hence $0.8D$), so their product is at least $(0.8D)^k$. Thus, if x is such that

$$\frac{x}{4D \log x} \geq \frac{\log M}{\log 0.8D} ,$$

then, we can always find sufficiently many distinct primes in the AP $A_{\tilde{D}}$ such that their product is at least M. This is true, for instance, for our choice of x above.

For the moreover part, we try all possible values of $\tilde{D}$ in the range $[0.8D, D]$ and for each of these choices and $x = (D \log M)^3$, we check if the arithmetic progression $A_{\tilde{D}}$ has sufficiently many primes using the deterministic primality test of Agrawal, Kayal, and Saxena[AKS04]. All these operations can be done deterministically in time $\mathsf{poly}(D, \log M)$.

□

4.2 Explicit Kakeya Sets of Higher Degree

We start with the definition of Kakeya sets of high degree.

Definition 4.4 ([BKW19])**.** *Let $\mathbb{F}$ be a finite field and let $u, m \in \mathbb{N}$. A set $K \subseteq \mathbb{F}^m$ is said to be a Kakeya set of degree u in $\mathbb{F}^m$ if there exist functions $g_0, g_1, \ldots, g_{u-1} : \mathbb{F}^m \to \mathbb{F}^m$ such that for every $\mathbf{a} \in \mathbb{F}^m$, the set of points*

$$\{g_0(\mathbf{a}) + g_1(\mathbf{a}) \cdot \tau + \cdots + g_{u-1}(\mathbf{a}) \cdot \tau^{u-1} + \mathbf{a} \cdot \tau^u : \tau \in \mathbb{F}\}$$

is a subset of K. ◊

For ease of notation, we denote the curve

$$\{g_0(\mathbf{a}) + g_1(\mathbf{a}) \cdot y + \cdots + g_{u-1}(\mathbf{a}) \cdot y^{u-1} + \mathbf{a} \cdot y^u : y \in \mathbb{F}\}$$

of degree u by $G_{\mathbf{a}}(y)$.

In their work [BKW19], Björklund, Kaski and Williams gave an explicit construction of Kakeya sets of degree u of non-trivially small size, provided that the degree u and the field size $\mathbb{F}$ satisfy an appropriate divisibility condition. This construction will be crucial for our algorithm.

Theorem 4.5 (Explicit Kakeya sets of degree u [BKW19])**.** *Let $\mathbb{F}$ be a finite field of size q, and let $u \in \mathbb{N}$ be such that $u+1$ divides $q-1$. Then, for every $m \in \mathbb{N}$, there is a Kakeya set K of degree u in $\mathbb{F}^m$ of size at most $\left(\frac{q-1}{u+1}+1\right)^m$.*

Moreover, this set K is a union of at most q product sets in $\mathbb{F}^m$ and there is a deterministic algorithm that on input $u, m, \mathbb{F}$, outputs K and the associated functions $g_0, g_1, \ldots, g_{u-1}$ in time $O(q|K|)$.

Proof. We start by restating the Kakeya set of degree u as described in [BKW19, Lemma 1].

$$K := \left\{ \left(\left(\frac{\alpha_1}{u+1} + \tau\right)^{u+1} - \tau^{u+1}, \ldots, \left(\frac{\alpha_m}{u+1} + \tau\right)^{u+1} - \tau^{u+1} \right) \Big| \, \forall\, \alpha_1, \alpha_2, \ldots, \alpha_m, \tau \in \mathbb{F}_q \right\}.$$

Observe that, $|K| \leq \left(\frac{q-1}{u+1}+1\right)^{m+1}$, as $|\{\beta^{u+1} : \beta \in \mathbb{F}_q\}| = \frac{q-1}{u+1}+1$. Also, the associated functions $g_0, g_1, \ldots, g_{u-1}$ can be computed in time $O(q|K|)$ by [BKW19, Lemma 1].

We now show that K is a union of at most q product sets. For $\tau \in \mathbb{F}_q$, define $S_\tau := \{\left(\frac{\alpha}{u+1} + \tau\right)^{u+1} - \tau^{u+1} \,|\, \forall \alpha \in \mathbb{F}_q\} \subseteq \mathbb{F}_q$. The proof concludes by observing that

$$K = \bigcup_{\tau \in \mathbb{F}_q} \underbrace{S_\tau \times S_\tau \times \cdots \times S_\tau}_{m \text{ times}}.$$

□

Using the property that the set K in Theorem 4.5 is a union of product sets and that for product sets we have nearly linear algorithms for multipoint evaluation using Lemma 3.10, we get the following.

Lemma 4.6. *Let $\mathbb{F}$ be a finite field of size q, $u \in \mathbb{N}$ be such that $u+1$ divides $q-1$, and $m \in \mathbb{N}$ be a natural number. Let K be the Kakeya set of degree u given by Theorem 4.5 over $\mathbb{F}^m$ and let $f(\mathbf{x})$ be a polynomial of degree less than d in each variable with coefficients in $\mathbb{F}$.*

Then, there is a deterministic algorithm that takes as input the set K and the coefficient vector of f and outputs the evaluation of f at every point in K in time

$$O(|K| + d^m) \cdot \mathsf{poly}(m, d, q)\,.$$

Proof. Recall K is a union of at most q product sets, $K = \bigcup_{\tau \in \mathbb{F}_q} \underbrace{S_\tau \times S_\tau \times \cdots \times S_\tau}_{m \text{ times}}$, here S_τ is as defined in proof of Theorem 4.5. For each $\tau \in \mathbb{F}_q$, by using Lemma 3.10, we can evaluate f on S_τ in time $(d^m + |S_\tau|^m) \cdot \mathsf{poly}(m, d, \log|\mathbb{F}|)$. Thus, we can evaluate f on every point in K in time $O(|K| + d^m) \cdot \mathsf{poly}(m, d, q)$. □

4.3 Fast Multipoint Evaluation over Nice Finite Fields

Theorem 4.7. *Let $\mathbb{F}$ be a finite field of size q and let $d, \tilde{d}, m \in \mathbb{N}$ be such that $\tilde{d} \in [0.8d, d]$ and $\tilde{d}-1$ divides $q-1$.*

Then, there is an algorithm that given a homogeneous m-variate polynomial in $\mathbb{F}[\mathbf{x}]$ of degree less than d in every variable and a set of N input points in $\mathbb{F}^m$, outputs the evaluation of this polynomial on these inputs in time

$$(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(q, m, d)\,.$$

Proof. Let f be the input polynomial and let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$ be the input points of interest.

At a high level, the algorithm here is similar in structure to that in [BGKM21]. We first evaluate the polynomial on an appropriate product set $\mathcal{P}$ in nearly linear time using Lemma 3.10 in the preprocessing phase. Next, in the local computation step, we look at the restriction of f on a curve $C_{\mathbf{a}}$ *through* any point $\mathbf{a} \in \mathbb{F}^m$ of interest. Based on the construction of the aforementioned product set $\mathcal{P}$, we will guarantee that there is a curve $C_{\mathbf{a}}$ through $\mathbf{a}$ such that the intersection of $C_{\mathbf{a}}$ with the set $\mathcal{P}$ is sufficiently large, so that the univariate polynomial obtained by restricting f to $C_{\mathbf{a}}$ can be uniquely *decoded* using the evaluation of f on $\mathcal{P}$. We then use this decoded polynomial to obtain $f(\mathbf{a})$.

Despite this high-level similarity, there are some technical differences between the algorithm here and that in [BGKM21]. Primarily, these differences arise due to the fact that unlike the setting in [BGKM21], we are no longer working over fields of small characteristic. So, the construction of the set $\mathcal{P}$ is different here and is based on the ideas in [BKW19]. We now specify the details, starting with the description of the algorithm.

The algorithm.

1. From the coefficient vector of f, compute each of Hasse derivatives of f of order at most $2m$.
2. Using Theorem 4.5, we construct a Kakeya set K of order $u = (q-1)/(\tilde{d}-1) - 1$. As is necessary, $u+1$ divides $q-1$. Note that, $|K| \leq \tilde{d}^{(m+1)}$.
3. For every Hasse derivative $\tilde{f}$ of f order at most $2m$, evaluate $\tilde{f}$ on K using Lemma 4.6.
4. For every $i \in [N]$:
   (a) We consider the univariate polynomial $R_i(y)$ obtained by the restriction of f on the curve $G_{\mathbf{a}_i}(y)$. This is a univariate polynomial of degree at most $(d-1)m \cdot (q-1)/(\tilde{d}-1) < 2m(q-1)$. Using Lemma 3.8, compute the evaluation of $R_i(y)$ and all its $\leq 2m$ order Hasse derivatives on $\mathbb{F}$.
   (b) Since degree of R_i is less than $2m(q-1)$, and we have the evaluation of R_i and all its derivatives of order at most $2m$ on q points, we can recover R_i uniquely from this information. In particular, we use Lemma 3.9 to recover $R_i(y)$.
   (c) We output $f(\mathbf{a}_i)$ to be equal to the leading coefficient of $R_i(y)$.

Running time. There are a total of $\binom{m+2m}{m}$ Hasse derivatives of order at most $2m$ of f. By Lemma 3.6, each Hasse derivative takes time $d^m \cdot \mathsf{poly}(m, d, \log q)$. Thus, the total time is bounded by $\Theta(d)^m \cdot \mathsf{poly}(m, d, \log q)$. By Theorem 4.5, we can compute the explicit Kakeya set of degree $u := \frac{q-1}{\tilde{d}-1} - 1$ in time $O(|K|q) = \Theta(d)^m \cdot q$. For the time complexity of Step 3, by Lemma 4.6, we can evaluate f and all its Hasse derivatives of f of order at most $2m$ in time

$$\binom{m+2m}{m} \cdot \left(\tilde{d}^{m+1} + d^m\right) \cdot \mathsf{poly}(m, d, q) = \Theta(d)^m \cdot \mathsf{poly}(m, d, q).$$

For the loop in Step 4, the time complexity is bounded by $\Theta(1)^m \cdot N \cdot \mathsf{poly}(d, q)$ by Lemma 3.8. By Hermite interpolation (Lemma 3.9), we can recover $R_i(y)$ for each i in time $\mathsf{poly}(m, q)$. Thus, the total time complexity is bounded by $(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(m, d, q)$.

Correctness. Note that, for any *homogenous* polynomial $f(\mathbf{x})$, the leading coefficient of $R_i(y) = f(g_0(\mathbf{a}_i) + g_1(\mathbf{a}_i) \cdot y + \cdots + g_{r-1}(\mathbf{a}_i) \cdot y^{r-1} + \mathbf{a} \cdot y^r)$ is $f(\mathbf{a}_i)$. Thus, given the polynomial $R_i(y)$, then we get $f(\mathbf{a}_i)$ by directly reading off its leading coefficient. Since the degree of $R_i(y)$ is $\leq 2m(q-1)$, via Hermite interpolation (Lemma 3.9) it suffices to know the evaluations of $\leq 2m$ order Hasse derivatives of $R_i(y)$ on $\mathbb{F}_q$. Finally, note that, we have access to evaluations of $\leq 2m$ order partial derivatives of $R_i(y)$ for all $i \in [N]$ because of Lemma 3.8 and Lemma 4.6. This concludes the proof. □

5 The First Algorithm over Rings of the Form $\mathbb{Z}/r\mathbb{Z}$

With the necessary background in place, we are now ready to describe our first algorithm for fast multivariate multipoint evaluation over rings of the form $\mathbb{Z}/r\mathbb{Z}$. This already handles the case of prime fields, and contains most of our main ideas. Later, in Section 6, we discuss the case of extension rings which will complete the proof of Theorem 1.1. Also, the algorithm in Section 6 crucially relies on the algorithm for the $\mathbb{Z}/r\mathbb{Z}$ case and an idea of Kedlaya and Umans [KU11].

5.1 The Description of the Algorithm

Algorithm 1 The First Algorithm over Rings of the Form $\mathbb{Z}/r\mathbb{Z}$

Algorithm MME-A$(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r)$

where f is an m-variate *homogeneous* polynomial over $\mathbb{Z}/r\mathbb{Z}$ of individual degree less than d and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in (\mathbb{Z}/r\mathbb{Z})^m$ are evaluation points.

1. Let $F \in \mathbb{Z}[\mathbf{x}]$ be the m-variate homogeneous polynomial of individual degree less than d obtained from f by replacing each of its coefficients with its natural lift in the set $[\![r]\!]$ of integers.

2. For every $i \in [N]$, let $\tilde{\mathbf{a}}_i \in [\![r]\!]^m$ be the lift of $\mathbf{a}_i \in (\mathbb{Z}/r\mathbb{Z})^m$ to the integers.

3. Let $M = d^m r^{dm}$. We invoke Lemma 4.3 with parameters d and M and obtain a natural number $\tilde{d} \in [0.8d, d]$ and primes $p_1, p_2, \ldots, p_k$ where $k \leq d^2(\log M)^3$, each $p_i \leq d^3(\log M)^3$ and is congruent to 1 modulo $\tilde{d}$, and $\prod_{i \in [k]} p_i > M$.

4. For $j \in [k]$, let $f_j(\mathbf{x}) \in \mathbb{F}_{p_i}[\mathbf{x}]$ be the m-variate homogeneous polynomial of individual degree less than d obtained by reducing each of the coefficients of F modulo the prime p_j. Similarly, for every $i \in [N]$, let $\mathbf{a}_{i,j} \in \mathbb{F}_{p_j}^m$ be obtained by reducing each of the coordinates of $\tilde{\mathbf{a}}_i$ modulo p_j.

5. For every $j \in [k]$, invoke the algorithm in Theorem 4.7 for the polynomial f_j, input points $\{\mathbf{a}_{i,j} : i \in [N]\}$ and parameters $d, \tilde{d}$ as above, and get $f_j(\mathbf{a}_{i,j})$ for all $j \in [k]$ and $i \in [N]$. Note that each f_j is a homogeneous polynomial, and from the guarantees of Lemma 4.3, $\tilde{d}$ is in the range $[0.8d, d]$ and $\tilde{d} - 1$ divides $p_i - 1$ as needed by Theorem 4.7.

6. For every $i \in [N]$, use the Chinese Remainder Theorem (Theorem 3.3) to compute $F(\tilde{\mathbf{a}}_i)$ from $\{f_j(\mathbf{a}_{i,j}) : j \in [k]\}$.

7. For every $i \in [N]$, output $f(\mathbf{a}_i) = F(\tilde{\mathbf{a}}_i) \mod r$.

5.2 The Correctness of Algorithm 1

The correctness of the algorithm essentially follows from the correctness of the building blocks in Section 4 and preliminary notions like Theorem 3.3. We now elaborate on some of the details.

In its first two steps, Algorithm 1 lifts the problem of multipoint evaluation given over the ring $\mathbb{Z}/r\mathbb{Z}$ to an instance over $\mathbb{Z}$ by naturally identifying the elements of $\mathbb{Z}/r\mathbb{Z}$ with the set $[\![r]\!]$ of integers. This lifted instance is given by the polynomial $F \in \mathbb{Z}[\mathbf{x}]$ and the points $\{\tilde{\mathbf{a}}_i : i \in [N]\}$. Thus, to correctly solve the original problem over $\mathbb{Z}/r\mathbb{Z}$, it suffices to correctly solve this lift over $\mathbb{Z}$ and then reduce the outputs modulo r, since for every $i \in [N]$, $f(\mathbf{a}_i) = F(\tilde{\mathbf{a}}_i) \mod r$. Hence, it suffices to argue that the computation of $F(\tilde{\mathbf{a}}_i)$ is correct for every $i \in [N]$.

Since F is an m-variate polynomial of degree less than d in each variable with every coefficient being in the set $[\![r]\!]$, and for every $i \in [N]$, each coordinate of $\tilde{\mathbf{a}}_i$ is also in the set $[\![r]\!]$, we get that $F(\tilde{\mathbf{a}}_i)$ is a natural number of absolute value at most $d^m(r-1)^{dm}$. Thus, for $M = d^m r^{dm} > d^m(r-1)^{dm}$, it suffices to compute $F(\tilde{\mathbf{a}}_i)$ modulo M.

This computation modulo M is done by working modulo distinct primes $p_1, p_2, \ldots, p_k$, each not too large, such that their product exceeds M. Moreover, these primes are carefully chosen using Lemma 4.3 which additionally ensures that there is a $\tilde{d} \in [0.8d, d]$ such that each of these primes is in the arithmetic progression $(1, 1+\tilde{d}, 1+2\tilde{d}, \ldots)$.

Given this additional structure on the primes, we next use Theorem 4.7 to solve the problem of evaluating the polynomial $f_j = F \mod p_j$ on inputs $\{\mathbf{a}_{i,j} : i \in [N]\}$, where $\mathbf{a}_{i,j} = \tilde{\mathbf{a}}_i \mod p_j$ over the field $\mathbb{F}_{p_j}$. Since f_j is homogeneous and the divisibility condition needed in the hypothesis of Theorem 4.7 holds, we have that this step works correctly and within the desired time guarantees. In other words, for every $i \in [N]$, the inputs to the Chinese Remainder Step (Step 6) of the algorithm are all correct. Thus, each of the evaluations of F and hence, those of f output by the algorithm are correct.

5.3 The Time Complexity of Algorithm 1

To bound the time complexity, we bound the time complexity of each of the steps of the algorithm.

The first two steps of the algorithm essentially require no additional computation beyond a reading of the input. We just semantically re-interpret the coefficients of f and the coordinates of $\mathbf{a}_i$ to be over the integers. Thus, these can be done in time $(d^m + N) \cdot \mathsf{poly}(\log r, m, d)$. From the choice of M in Step 3 and the guarantees in Lemma 4.3, it follows that the time complexity of Step 3 is at most $\mathsf{poly}(\log M, d) = \mathsf{poly}(d, m, \log r)$. Step 4 again takes at most $(d^m + N) \cdot \mathsf{poly}(d, m, \log r)$ time since each of the primes p_i has absolute value at most $\mathsf{poly}(d, m, \log r)$ from Lemma 4.3. For every $j \in [k]$, it follows from Theorem 4.7 that multipoint evaluation of the polynomial f_j on inputs $\{\mathbf{a}_{i,j} : i \in [N]\} \subseteq \mathbb{F}_{p_j}^m$ takes at most $(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(d, m, p_j) = (d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(d, m, \log r)$ from the bound on the magnitude of p_j. However, we note that to use Theorem 4.7, we need the inputs in this function call to satisfy the divisibility condition in the hypothesis of the theorem, namely that $\tilde{d} - 1$ divides $p_j - 1$. But we have already argued this while arguing the correctness of the algorithm.

Step 6 is a straightforward application of the Chinese Remainder Theorem and using Theorem 3.3, we can do this in time at most $N \cdot \mathsf{poly}(k, \max_j(\log p_j)) \leq N \cdot \mathsf{poly}(d, m, \log r)$. Finally, the last step is just a sequence of N integer divisions involving numbers of bit complexity at most $\log M$, and hence can be implemented in time $N \cdot \mathsf{poly}(\log M) = N \cdot \mathsf{poly}(d, m, \log r)$.

Thus, the overall time complexity of the algorithm is at most $(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(d, m, \log r)$.

We summarize the above discussion in the following theorem.

Theorem 5.1. *Let $f(\mathbf{x})$ be a homogeneous m-variate polynomial over $\mathbb{Z}/r\mathbb{Z}$ of individual degree less than d. Let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ be N points from $(\mathbb{Z}/r\mathbb{Z})^m$. Then, given $(f, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r)$ as the input to Algorithm 1, it computes $f(\mathbf{a}_i)$ for all $i \in [N]$ in time*

$$(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(m, d, \log r).$$

6 The First Algorithm over Extension Rings

In this section, we extend the fast multivariate multipoint evaluation algorithm discussed in the previous section to over extension rings of the form $(\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ for a polynomial $E(z) \in \mathbb{Z}/r\mathbb{Z}[z]$, and in particular over all finite fields. This will prove our Theorem 1.1. Here, the underlying ring is of the form $(\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ where $E(z)$ is a monic polynomial of degree at most $e-1$. The idea for this extension is essentially the same as that used by Kedlaya and Umans in [KU11, §4.3] to extend their algorithm over rings of the form $\mathbb{Z}/r\mathbb{Z}$ to other extension rings. They essentially reduce an instance of multivariate multipoint evaluation over extension rings to an instance over rings of the form $\mathbb{Z}/r'\mathbb{Z}$. Then, they invoke their multivariate multipoint evaluation algorithm over $\mathbb{Z}/r'\mathbb{Z}$. For our proof, we use our Algorithm 1 for solving the problem over $\mathbb{Z}/r'\mathbb{Z}$. The improved dependence on the number of variables for our algorithm thus just follows from the improved dependence on the number of variables in the complexity of Algorithm 1. The rest of the steps are the same as [KU11]. We now describe the steps of the algorithm.

6.1 The Description of the Algorithm

Algorithm 2 The First Algorithm over Extension Rings

Algorithm MME-FOR-EXTENSION-RINGS-A$(f(\mathbf{x}), \mathbf{a}_1, \ldots, \mathbf{a}_N)$

where R is the underlying ring represented as $(\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ such that $E(z)$ is a degree e monic polynomial over $\mathbb{Z}/r\mathbb{Z}$, $f(\mathbf{x})$ is an m-variate *homogeneous* polynomial over R of individual degree less than d, and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ are points in R^m.

Let $M := d^m(e(r-1))^{(d-1)m+1} + 1$ and $r' := M^{(e-1)dm+1}$.

1. Let $F(\mathbf{x}) \in \mathbb{Z}[z][\mathbf{x}]$ be the m-variate homogeneous polynomial of individual degree less than d obtained by replacing each coefficient of $f(\mathbf{x})$ (which is a polynomial in $\mathbb{Z}/r\mathbb{Z}[z]$) with its natural lift to a polynomial in z over the integers by identifying $\mathbb{Z}/r\mathbb{Z}$ with the set of integers $[\![r]\!]$. Similarly, for every $i \in [N]$, let $\tilde{\mathbf{a}}_i \in \mathbb{Z}[z]^m$ be the lift of the point $\mathbf{a}_i \in R^m$.

2. Let $\overline{f}(\mathbf{x})$ be the polynomial computed from $F(\mathbf{x})$ by reducing its coefficients (which are elements of $\mathbb{Z}[z]$) modulo $z - M$ and r', i.e., for each of these polynomials in $\mathbb{Z}[z]$, we first set $z = M$ and then reduce the result modulo r'. Similarly, for all $i \in [N]$, $\overline{\mathbf{a}}_i$ is the point obtained from $\tilde{\mathbf{a}}_i$ by reducing each of its coordinates modulo $z - M$ and r'.

   Note that from the choice of r', going modulo r' does not change anything computationally, but formally reduces the problem to an instance of multivariate multipoint evaluation over $\mathbb{Z}/r'\mathbb{Z}$.

3. Call the function MME-A$(\overline{f}, \overline{\mathbf{a}}_1, \overline{\mathbf{a}}_2, \ldots, \overline{\mathbf{a}}_N, r'))$ in Algorithm 1 and get $\overline{b}_i = \overline{f}(\overline{\mathbf{a}}_i)$ for all $i \in [N]$.

4. For all $i \in [N]$, from $\overline{b}_i$ compute the unique polynomial $Q_i(z)$ in $\mathbb{Z}[z]$ of degree at most $(e-1)dm$ and coefficients are in $[\![M]\!]$ such that $Q_i(M)$ is congruent to $\overline{b_i}$ modulo r'. In other words, we compute base M representation of the natural number $\overline{b}_i$. Finally, we compute $Q_i(z)$ modulo r and $E(z)$ and get $f(\mathbf{a}_i)$ for all $i \in [N]$.

6.2 The Correctness of Algorithm 2

We show that Algorithm 2 successfully computes $f(\mathbf{a}_i)$ for all $i \in [N]$. From the first step of the above algorithm, we observe that for all $i \in [N]$, $f(\mathbf{a}_i)$ equals $F(\tilde{\mathbf{a}}_i)$ modulo r and $E(z)$. Therefore, at the end of step 1, Algorithm 2 reduces the problem of computing $f(\mathbf{a}_i)$ for all $i \in [N]$ over the extension ring to the problem of computing $F(\tilde{\mathbf{a}}_i)$ for all $i \in [N]$ over integers. The next natural step is to solve this problem of computing $F(\tilde{\mathbf{a}}_i)$ for all $i \in [N]$ using Algorithm 1. To this end, we further reduce this problem of computing $F(\tilde{\mathbf{a}}_i)$ for $i \in [N]$ to an instance of multivariate multipoint evaluation over rings of the form $\mathbb{Z}/r'\mathbb{Z}$.

From the construction of F, we know that the coefficients of $F(\mathbf{x})$ and the coordinates of $\tilde{\mathbf{a}}_i$ are all polynomials in z of degree at most $e-1$, and the coefficients of these polynomials are integers in the set $[\![r]\!]$. Thus, $F(\tilde{\mathbf{a}}_i)$ is a polynomial in z of degree at most $(e-1)dm$ and each of its coefficients is a non-negative integer of absolute value at most $d^m \cdot (e(r-1))^{(d-1)m} \cdot (r-1) \leq d^m \cdot (e(r-1))^{(d-1)m+1}$, which by our choice of M is at most $M-1$. Recall that an arbitrary polynomial $P \in \mathbb{Z}[z]$ with non-negative integer coefficients of absolute value less than M can be uniquely (and efficiently) recovered given its evaluation at M: just construct the base M representation of $P(M)$ (or equivalently $P(z) \mod (z-M)$), and read off the digits of such a representation. Based on this, we set ourselves the goal of computing $F(\tilde{\mathbf{a}}_i)$ by computing $F(\tilde{\mathbf{a}}_i) \mod (z-M)$. From the choice of r', r' is strictly larger than the integer $F(\tilde{\mathbf{a}}_i) \mod (z-M)$ and hence it suffices to compute $F(\tilde{\mathbf{a}}_i) \mod (z-M)$ while working modulo r'. Note that this is equal to $\overline{f}(\overline{\mathbf{a}}_i)$ in our notation. This reduction is done in step 2 of the above algorithm and reduces the original problem to an instance of multipoint evaluation over $\mathbb{Z}/r'\mathbb{Z}$. This computation, in turn is done in step 3 of the algorithm using Algorithm 1. The output of this step is $\overline{b}_i = \overline{f}(\overline{\mathbf{a}}_i)$ for all $i \in [N]$. From $\overline{b}_i$, we get $F(\tilde{\mathbf{a}}_i)$ (which is the same as $Q_i(z)$ in the algorithm) by computing the representation of the number $\overline{b}_i$ with respect to the base M. From this, we get $f(\mathbf{a}_i)$ easily by reducing $Q_i(z)$ modulo r and $E(z)$. This completes the proof of correctness of the algorithm.

6.3 The Time Complexity of Algorithm 2

In step 1 of Algorithm 2, the cost of lifting the polynomial $f(\mathbf{x})$ and the points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ to over integers is $O((d^m + mN)e \log r)$. Step 2 of the algorithm computes the polynomial $\overline{f}(\mathbf{x})$ which is $F(\mathbf{x})$ modulo $z-M$ and r'. Computing $F(\mathbf{x})$ modulo $z-M$ is the same as evaluating each of its coefficients at $z = M$. Since each coefficient of $F(\mathbf{x})$ is a polynomial in z of degree at most $e-1$ and coefficients are in the set of integers $[\![r]\!]$, evaluating each coefficient of $F(\mathbf{x})$ at $z = M$ can be done in time $e \cdot \mathsf{poly}(\log r, \log M)$ which is at most $\mathsf{poly}(d, m, e, \log r)$. Also, note that computationally reduction modulo r' does not involve any actual computation since r' is very large and hence the cost of computing $\overline{f}(\mathbf{x})$ is $d^m \cdot \mathsf{poly}(d, m, e, \log r)$. Similarly, computing $\overline{\mathbf{a}}_i$ for all $i \in [N]$ can be done in time $N \cdot \mathsf{poly}(d, m, e, \log r)$. Thus, the cost of the step 2 of Algorithm 2 is $(d^m + N) \cdot \mathsf{poly}(d, m, e, \log r)$. Step 3 of the algorithm invokes the function MME-A$(\overline{f}, \overline{\mathbf{a}}_1, \overline{\mathbf{a}}_2, \ldots, \overline{\mathbf{a}}_N, r')$ in Algorithm 1 and it runs in time $(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(d, m, e, \log r)$ (see Theorem 5.1). In step 4, for each $i \in [N]$, computing the polynomial $Q_i[z]$ from $\overline{b}_i$ is the same as computing a base M representation of the integer $\overline{b}_i$ of absolute value at most $r' \leq \exp(\mathsf{poly}(e, d, m, \log r))$. This can be done in time $\mathsf{poly}(d, m, e, \log r)$. From $Q_i[z]$, computing $f(\mathbf{a}_i)$ involves reduction modulo r and $E(z)$ and this takes time $\mathsf{poly}(d, m, e, \log r)$. Hence, the total cost of the step 4 is $N \cdot \mathsf{poly}(d, m, e, \log r)$.

Thus, the overall time complexity of Algorithm 2 is at most $(d^m + N) \cdot \Theta(1)^m \cdot \mathsf{poly}(d, m, e, \log r)$.

6.4 Proof of Theorem 1.1

Now, we describe the proof of Theorem 1.1.

Proof of Theorem 1.1. Let p be the characteristic of the underlying finite field $\mathbb{F}$ and $|\mathbb{F}| = p^e$. Then, we can assume that $\mathbb{F}$ is represented by $(\mathbb{Z}/p\mathbb{Z})[z]/(E(z))$ for some irreducible monic polynomial over $\mathbb{Z}/p\mathbb{Z}$. Let $f(\mathbf{x})$ be the input polynomial with m variables and degree less than d in each variable, and $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N \in \mathbb{F}^m$ be the input points. An m-variate polynomial of

individual degree less than d can have at most $(d-1)m+1$ many homogeneous components. For all $j \in [\![(d-1)m+1]\!]$, let f_j be the degree j homogeneous component of f. Each f_j can be computed in time $d^m \cdot \mathsf{poly}(d,m)$ by counting the degree of each monomial in $f(\mathbf{x})$. Therefore, the total cost of computing all f_j's is $d^m \cdot \mathsf{poly}(d,m)$. Now for each $j \in [\![(d-1)m+1]\!]$, we invoke Algorithm 2 with input $(f_j(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N)$ and compute $f_j(\mathbf{a}_i)$ for all $i \in [N]$ in time $(d^m+N)\cdot\Theta(1)^m \cdot \mathsf{poly}(m,d,\log|\mathbb{F}|)$. Hence, the total cost of this step is bounded by $(d^m+N)\cdot\Theta(1)^m \cdot \mathsf{poly}(m,d,\log|\mathbb{F}|)$. Note that $f(\mathbf{a}_i) = \sum_{j=0}^{(d-1)m} f_j(\mathbf{a}_i)$. This implies that for each $i \in [N]$, the cost of computing $f(\mathbf{a}_i)$ from $f_j(\mathbf{a}_i)$'s is $\mathsf{poly}(m,d,\log|\mathbb{F}|)$. Therefore, the total cost incurred by this step is $N\cdot\mathsf{poly}(m,d,\log|\mathbb{F}|)$. Thus, the total time taken to compute $f(\mathbf{a}_i)$ for all $i \in [N]$ is $(d^m+N)\cdot\Theta(1)^m\cdot\mathsf{poly}(m,d,\log|\mathbb{F}|)$ which is $(d^m+N)^{1+o(1)}\cdot\mathsf{poly}(m,d,\log|\mathbb{F}|)$. □

7 The Second Algorithm over Rings of the Form $\mathbb{Z}/r\mathbb{Z}$

The main result of this section is the following theorem.

Theorem 7.1. *Over $\mathbb{Z}/r\mathbb{Z}$, for all $m \in \mathbb{N}$ and sufficiently large $d \in \mathbb{N}$, there exists a deterministic algorithm that outputs the evaluation of an m-variate polynomial of degree less than d in each variable on N points in time*

$$(d^m+N)^{1+o(1)} \cdot \mathsf{poly}(m,d,\log r),$$

provided that $\log^{\circ c} r \le d^{o(1)}$ for some fixed constant $c \in \mathbb{N}$.

We need the following lemma. It gives a way of computing the evaluation of $f(\mathbf{x})$ over a ring R at a point $\mathbf{a}$ from the evaluations of Hasse derivatives of $f(\mathbf{x})$ at another point $\mathbf{b}$, provided that the coordinates of $\mathbf{a}-\mathbf{b}$ are in a nilpotent ideal of R.

Lemma 7.2. *Let $f(\mathbf{x})$ be an m-variate polynomial over a commutative ring R. Let I be an ideal of R and s be a positive integer such that $I^s = 0$. Let $\mathbf{a} = (a_1,\ldots,a_m), \mathbf{b} = (b_1,\ldots,b_m) \in R^m$ such that $a_i \equiv b_i \pmod{I}$ for $i \in [m]$. Then*

$$f(\mathbf{a}) = \sum_{\mathbf{e}\in\mathbb{N}^m : |\mathbf{e}|_1 < s} \overline{\partial}_{\mathbf{e}}(f)(\mathbf{b}) \cdot (\mathbf{a}-\mathbf{b})^{\mathbf{e}}.$$

Proof. By the definition of Hasse derivatives,

$$f(\mathbf{a}) = f(\mathbf{b}+(\mathbf{a}-\mathbf{b})) = \sum_{\mathbf{e}\in\mathbb{N}^m} \overline{\partial}_{\mathbf{e}}(f)(\mathbf{b}) \cdot (\mathbf{a}-\mathbf{b})^{\mathbf{e}}.$$

The above sum is well-defined since $\overline{\partial}_{\mathbf{e}}(f) = 0$ when $|\mathbf{e}|_1$ is sufficiently large. The lemma follows by noting that $(\mathbf{a}-\mathbf{b})^{\mathbf{e}}$ is in the ideal $I^{|\mathbf{e}|_1}$, which is zero when $|\mathbf{e}|_1 \ge s$. □

7.1 A Basic Algorithm

We first describe a basic algorithm, MME-Product-Set, that evaluates a polynomial $f(\mathbf{x}) \in (\mathbb{Z}/r^s\mathbb{Z})[\mathbf{x}]$ at N points in $(\mathbb{Z}/r^s\mathbb{Z})^m$ simultaneously. Its time complexity is not good enough, but this will be improved in later subsections.

Algorithm 3 Basic Algorithm

Algorithm MME-Product-Set$(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s)$

where $f(\mathbf{x})$ is an m-variate polynomial over $\mathbb{Z}/r^s\mathbb{Z}$ of individual degree at most $d-1$, $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N$ are evaluation points in $(\mathbb{Z}/r^s\mathbb{Z})^m$, and $s \in [m]$.

1. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$, use Lemma 3.6 to compute $f_{\mathbf{e}}(\mathbf{x}) := \overline{\partial}_{\mathbf{e}}(f)(\mathbf{x}) \in (\mathbb{Z}/r^s\mathbb{Z})[\mathbf{x}]$.

2. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$, use Lemma 3.10 to compute $f_{\mathbf{e}}(\mathbf{a}) \in \mathbb{Z}/r^s\mathbb{Z}$ for $\mathbf{a} \in [\![r]\!]^m$, where $[\![r]\!]$ is identified with a subset of $\mathbb{Z}/r^s\mathbb{Z}$ via $i \mapsto i + r^s\mathbb{Z}$.

3. For all $i \in [N]$, compute $\bar{\mathbf{a}}_i \in [\![r]\!]^m \subseteq (\mathbb{Z}/r^s\mathbb{Z})^m$ such that the coordinates of $\bar{\mathbf{a}}_i$ are the remainders of the corresponding coordinates of $\mathbf{a}_i$ modulo r.

4. For all $i \in [N]$, compute $f(\mathbf{a}_i)$ by

$$f(\mathbf{a}_i) = \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < s} f_{\mathbf{e}}(\bar{\mathbf{a}}_i) \cdot (\mathbf{a}_i - \bar{\mathbf{a}}_i)^{\mathbf{e}} \in \mathbb{Z}/r^s\mathbb{Z} \tag{7.3}$$

   and output it.

Lemma 7.4. *Given the input $(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s)$, the algorithm* MME-Product-Set *computes $f(\mathbf{a}_i)$ for all $i \in [N]$ in time $O(\binom{m+s-1}{s-1}(d^m + r^m + N)) \cdot \mathsf{poly}(m, d, \log r)$.*

Proof. Let $I = r\mathbb{Z}/r^s\mathbb{Z}$. Then $I^s = 0$ and the coordinates of $\mathbf{a}_i - \bar{\mathbf{a}}_i$ are all in I. So (7.3) holds by Lemma 7.2. This shows that the algorithm correctly computes $f(\mathbf{a}_i)$ for $i \in [N]$.

Step 1 takes time $O(\binom{m+s-1}{s-1}d^m) \cdot \mathsf{poly}(m, d, \log r)$ by Lemma 3.6. And Step 2 takes time $O(\binom{m+s-1}{s-1}(d^m + r^m)) \cdot \mathsf{poly}(m, d, \log r)$ by Lemma 3.10. Step 3 takes time $O(N) \cdot \mathsf{poly}(m, \log r)$. Finally, Step 4 takes time $O(\binom{m+s-1}{s-1}N) \cdot \mathsf{poly}(m, \log r)$. So the total time complexity is $O(\binom{m+s-1}{s-1}(d^m + r^m + N)) \cdot \mathsf{poly}(m, d, \log r)$. □

7.2 The Description of the Algorithm

We describe the second algorithm MME-B now.

Algorithm 4 The Second Algorithm over $\mathbb{Z}/r^s\mathbb{Z}$

Algorithm MME-B$(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s, t)$

where $f(\mathbf{x})$ is an m-variate polynomial over $\mathbb{Z}/r^s\mathbb{Z}$ of individual degree at most $d-1$, $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N$ are evaluation points in $(\mathbb{Z}/r^s\mathbb{Z})^m$, $s \in [m]$, and $t \geq 0$ is the depth of the reduction tree.

1. If $t = 0$, invoke MME-Product-Set with input $(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s)$ to compute $f(\mathbf{a}_i)$ for $i \in [N]$, and return.
2. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$, use Lemma 3.6 to compute $f_{\mathbf{e}}(\mathbf{x}) := \overline{\partial}_{\mathbf{e}}(f)(\mathbf{x}) \in (\mathbb{Z}/r^s\mathbb{Z})[\mathbf{x}]$, and then compute a lift $\tilde{f}_{\mathbf{e}}(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ of $f_{\mathbf{e}}(\mathbf{x})$ with coefficients in $[\![r^s]\!]$.
3. For all $i \in [N]$, compute $\tilde{\mathbf{a}}_i \in [\![r]\!]^m$ such that the coordinates of $\tilde{\mathbf{a}}_i$ are the remainders of the corresponding coordinates of $\mathbf{a}_i$ modulo r, and compute $\bar{\mathbf{a}}_i := \tilde{\mathbf{a}}_i \bmod r^s \in (\mathbb{Z}/r^s\mathbb{Z})^m$.
4. Let $M := d(r-1)^d$. Find primes $p_1 < p_2 < \cdots < p_k$ less than or equal to $16 \log M$ such that $\prod_{j=1}^k p_j > M$.
5. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$ and $j \in [k]$, compute $f_{\mathbf{e},j}(\mathbf{x}) := \tilde{f}_{\mathbf{e}}(\mathbf{x}) \bmod p_j^m \in (\mathbb{Z}/p_j^m\mathbb{Z})[\mathbf{x}]$.
6. For all $i \in [N]$ and $j \in [k]$, compute $\mathbf{a}_{i,j} := \tilde{\mathbf{a}}_i \bmod p_j^m \in (\mathbb{Z}/p_j^m\mathbb{Z})^m$.
7. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$ and $j \in [k]$, invoke MME-B with input $(f_{\mathbf{e},j}, \mathbf{a}_{1,j}, \mathbf{a}_{2,j}, \ldots, \mathbf{a}_{N,j}, p_j, m, t-1)$ to compute $f_{\mathbf{e},j}(\mathbf{a}_{i,j}) \in \mathbb{Z}/p_j^m\mathbb{Z}$ for $i \in [N]$.
8. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < s$ and $i \in [N]$, use the Chinese Remainder Theorem (Theorem 3.3) to compute $\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i)$ as the unique $Q_i \in \left[\!\!\left[\prod_{j=1}^k p_j^m\right]\!\!\right]$ such that $Q_i \bmod p_j^m = f_{\mathbf{e},j}(\mathbf{a}_{i,j})$ for $j \in [k]$, and then compute $f_{\mathbf{e}}(\bar{\mathbf{a}}_i) = \tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) \bmod r^s \in \mathbb{Z}/r^s\mathbb{Z}$.
9. For all $i \in [N]$, compute and output

$$f(\mathbf{a}_i) = \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < s} f_{\mathbf{e}}(\bar{\mathbf{a}}_i) \cdot (\mathbf{a}_i - \bar{\mathbf{a}}_i)^{\mathbf{e}} \in \mathbb{Z}/r^s\mathbb{Z}. \tag{7.5}$$

7.3 The Correctness of Algorithm 4

We prove the correctness of the algorithm MME-B (Algorithm 4), as stated by the following claim.

Claim 7.6. *Given the input* $(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s, t)$*, the algorithm* MME-B *computes* $f(\mathbf{a}_i)$ *for all* $i \in [N]$.

Proof. We prove the claim via induction on t. When $t = 0$, the algorithm invokes MME-Product-Set in Step 1 to compute $f(\mathbf{a}_i)$ for $i \in [N]$ and the claim holds by Lemma 7.4.

Now consider $t \geq 1$ and assume the claim holds for $t' = t - 1$.

In Step 2, we compute the Hasse derivatives $f_{\mathbf{e}}(\mathbf{x}) = \overline{\partial}_{\mathbf{e}}(f)(\mathbf{x})$ of $f(\mathbf{x})$ and then lift them to $\tilde{f}_{\mathbf{e}}(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$. In Step 3, we compute $\tilde{\mathbf{a}}_i \in [\![r]\!]^m$ from $\mathbf{a}_i$ and then compute $\bar{\mathbf{a}}_i := \tilde{\mathbf{a}}_i \bmod r^s \in$

$(\mathbb{Z}/r^s\mathbb{Z})^m$. In Step 4, we compute the primes p_j, whose existence follows from Lemma 3.2. In Step 5 and Step 6, we compute $f_{\mathbf{e},j}(\mathbf{x}) := \tilde{f}_{\mathbf{e}}(\mathbf{x}) \bmod p_j^m$ and $\mathbf{a}_{i,j} := \tilde{\mathbf{a}}_i \bmod p_j^m$ respectively.

In Step 7, we invoke MME-B with input $(f_{\mathbf{e},j}, \mathbf{a}_{1,j}, \mathbf{a}_{2,j}, \ldots, \mathbf{a}_{N,j}, p_j, m, t')$ where $t' = t-1$. By the induction hypothesis, this correctly returns $f_{\mathbf{e},j}(\mathbf{a}_{i,j}) \in \mathbb{Z}/p_j^m\mathbb{Z}$ for $i \in [N]$.

In Step 8, we use the Chinese Remainder Theorem (Theorem 3.3) to compute the unique $Q_i \in \left[\!\left[\prod_{j=1}^k p_j^m\right]\!\right]$ for $i \in [N]$ such that

$$Q_i \bmod p_j^m = f_{\mathbf{e},j}(\mathbf{a}_{i,j}) \quad \text{for } j \in [k].$$

We claim $\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) = Q_i$. As $f_{\mathbf{e},j}(\mathbf{x}) = \tilde{f}_{\mathbf{e}}(\mathbf{x}) \bmod p_j^m$ and $\mathbf{a}_{i,j} := \tilde{\mathbf{a}}_i \bmod p_j^m$, we do have

$$\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) \bmod p_j^m = f_{\mathbf{e},j}(\mathbf{a}_{i,j}) \quad \text{for } j \in [k].$$

To prove the claim, it remains to show that $\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) \in \left[\!\left[\prod_{j=1}^k p_j^m\right]\!\right]$. As $\tilde{\mathbf{a}}_i \in [\![r]\!]^m$, $\tilde{f}_{\mathbf{e}}(\mathbf{x})$ is a polynomial of total degree at most $(d-1)m$ with at most d^m monomials, and the coefficients of these monomials are in $[\![r^s]\!]$, we see that $\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i)$ is a non-negative integer bounded by $d^m \cdot (r^s-1) \cdot (r-1)^{(d-1)m} < d^m r^{dm} < \prod_{j=1}^k p_j^m$. Here we use the facts $s \le m$ and $\prod_{j=1}^k p_j > M = d(r-1)^d$. So $\tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) = Q_i$.

We also compute $f_{\mathbf{e}}(\bar{\mathbf{a}}_i) = \tilde{f}_{\mathbf{e}}(\tilde{\mathbf{a}}_i) \bmod r^s \in \mathbb{Z}/r^s\mathbb{Z}$ in Step 8. This equality holds since $\tilde{f}_{\mathbf{e}}(\mathbf{x}) \bmod r^s = f_{\mathbf{e}}(\mathbf{x})$ and $\tilde{\mathbf{a}}_i \bmod r^s = \bar{\mathbf{a}}_i$.

Finally, in Step 9, we compute $f(\mathbf{a}_i)$ from the evaluations $f_{\mathbf{e}}(\bar{\mathbf{a}}_i)$ for $i \in [N]$ using (7.5). Let $I = r\mathbb{Z}/r^s\mathbb{Z}$. Then $I^s = 0$ and the coordinates of $\mathbf{a}_i - \bar{\mathbf{a}}_i$ are all in I by the choice of $\bar{\mathbf{a}}_i$. So (7.3) holds by Lemma 7.2. This proves the correctness of the algorithm. □

7.4 The Time Complexity of Algorithm 4

We now analyze the time complexity of Algorithm 4. When $t = 0$, the algorithm only executes Step 1, and its time complexity is $O(\binom{m+s-1}{s-1}(d^m + r^m + N)) \cdot \mathsf{poly}(m, d, \log r)$ by Lemma 7.4.

Now assume $t \ge 1$. Step 2 takes time $O(\binom{m+s-1}{s-1}d^m) \cdot \mathsf{poly}(m, d, \log r)$ by Lemma 3.6. Step 3 takes time $O(N) \cdot \mathsf{poly}(m, \log r)$. In Step 4, we compute the primes p_j using the Sieve of Eratosthenes [Sho08, §5.4], which takes time $\tilde{O}(\log M) \le \mathsf{poly}(d, \log r)$.

Using $k \le p_k = O(\log M) = O(d \log r)$, we see that Step 5 takes time $O(\binom{m+s-1}{s-1}d^m) \cdot \mathsf{poly}(m, d, \log r)$, and Step 6 takes time $O(N) \cdot \mathsf{poly}(m, d, \log r)$. Step 8 takes time $O(\binom{m+s-1}{s-1}N) \cdot \mathsf{poly}(m, d, \log r)$ by Theorem 3.3. Step 9 takes time $O(\binom{m+s-1}{s-1}N) \cdot \mathsf{poly}(m, d, \log r)$.

So the total time complexity of Steps 2–6 and 8–9 is

$$O\left(\binom{m+s-1}{s-1}(d^m + N)\right) \cdot \mathsf{poly}(m, d, \log r).$$

Let $T(r, s, t)$ be the time complexity of the algorithm. We have

$$T(r, s, 0) = O\left(\binom{m+s-1}{s-1}(d^m + r^m + N)\right) \cdot \mathsf{poly}(m, d, \log r). \tag{7.7}$$

And for $t \ge 1$,

$$T(r, s, t) \le O\left(\binom{m+s-1}{s-1}\log M\right) T(r', m, t-1) + O\left(\binom{m+s-1}{s-1}(d^m + N)\right) \mathsf{poly}(m, d, \log r). \tag{7.8}$$

where $r' = O(\log M) = O(d \log r)$.

For convenience, we define the function

$$\lambda_k(x) := \prod_{i=0}^{\min\{k, \log^\star x - 1\}} \log^{\circ i} x.$$

Observe that $\lambda_k(x) = x^{1+o(1)}$ and that $\lambda_i(x) \le \lambda_j(x)$ for $i \le j$ and $x \ge 1$.

Claim 7.9. *Let $r_0 = r \geq 2$ and $r_i = cd \log r_{i-1}$ for $i \geq 1$, where $c \geq 1$ is a constant. Then $r_k \leq c' \lambda_k(d) \log^{\circ k} r$ for all $0 \leq k < \log^\star r$ and a sufficiently large constant $c' \geq 1$.*

Proof. Induct on k. The claim holds obviously holds for $k = 0$. Now let $k \geq 1$. Assume the claim holds for $k' = k - 1$. Then

$$r_k = cd \log r_{k-1} \leq cd \log(c' \lambda_{k-1}(d) \log^{\circ(k-1)} r) = cd \log c' + cd \log(\lambda_{k-1}(d)) + cd \log^{\circ k} r.$$

As c' is sufficiently large, we may assume $cd \log c' + (c'/2)\lambda_k(d) + cd \log^{\circ k} r \leq c' \lambda_k(d) \log^{\circ k} r$. It remains to prove $cd \log(\lambda_{k-1}(d)) \leq (c'/2)\lambda_k(d)$. Here

$$\log(\lambda_{k-1}(d)) = \sum_{i=1}^{\min\{k, \log^\star d\}} \log^{\circ i} d \quad \text{and} \quad \lambda_k(d)/d = \prod_{i=1}^{\min\{k, \log^\star d - 1\}} \log^{\circ i} d$$

Repeatedly applying the fact that $ab \geq a + b$ when $a, b \geq 2$ shows that $c \log(\lambda_{k-1}(d)) \leq (c'/2)\lambda_k(d)/d$ for a sufficiently large constant c', and hence $cd \log(\lambda_{k-1}(d)) \leq (c'/2)\lambda_k(d)$. □

By Claim 7.9, at the k-th level of the recursion tree (where the top level is regarded as the first level), we have $\log M = O(\lambda_k(d) \log^{\circ k} r) \leq O(\lambda_t(d) \log^{\circ k} r)$. Using this to solve the recurrence relations (7.7) and (7.8), and noting that $\binom{m+s-1}{s-1} \leq 2^{O(m)}$ for $s \leq m$ (see Proposition 3.1), we obtain

$$T(r, s, t) = O\left(\binom{m+s-1}{s-1} 2^{Ctm} \lambda_t(d)^t \lambda_{t-1}(\log r) \left((\lambda_t(d) \log^{\circ t} r)^m + N\right)\right) \cdot \mathsf{poly}(m, d, \log r)$$

for $0 \leq t < \log^\star(r)$, where C is a large enough absolute constant.

Combining the above time complexity analysis with Claim 7.6 yields the following theorem.

Theorem 7.10. *Let $f(\mathbf{x})$ be an m-variate polynomial over $\mathbb{Z}/r^s\mathbb{Z}$ of individual degree less than d, where $s \leq m$. Let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ be N points in $(\mathbb{Z}/r\mathbb{Z})^m$. Let t be a non-negative integer less than $\log^\star r$. Then, given $(f, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, s, t)$, the algorithm* MME-B *computes $f(\mathbf{a}_i)$ for all $i \in [N]$ in time*

$$O\left(\binom{m+s-1}{s-1} 2^{Ctm} \lambda_t(d)^t \lambda_{t-1}(\log r) \left((\lambda_t(d) \log^{\circ t} r)^m + N\right)\right) \cdot \mathsf{poly}(m, d, \log r).$$

where $C > 0$ is a large enough absolute constant.

Now we are ready to prove Theorem 7.1. For convenience, we restate the theorem.

Theorem 7.1. *Over $\mathbb{Z}/r\mathbb{Z}$, for all $m \in \mathbb{N}$ and sufficiently large $d \in \mathbb{N}$, there exists a deterministic algorithm that outputs the evaluation of an m-variate polynomial of degree less than d in each variable on N points in time*

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log r),$$

provided that $\log^{\circ c} r \leq d^{o(1)}$ for some fixed constant $c \in \mathbb{N}$.

Proof. We invoke the algorithm MME-B for $t = \min\{c, \log^\star r - 1\}$ and $s = 1$. Then, from Theorem 7.10, we get the evaluations in time

$$O\left(2^{Ctm} \lambda_t(d)^t \lambda_{t-1}(\log r)((\lambda_t(d) \log^{\circ t} r)^m + N)\right) \cdot \mathsf{poly}(m, d, \log r).$$

where $C > 0$ is a large enough absolute constant. From the definition, $\lambda_t(d) = d^{1+o(1)}$. Also, $\log^{\circ t} r = d^{o(1)}$. Therefore, $\lambda_t(d) \log^{\circ t} r = d^{1+o(1)}$. Since t is bounded by some constant, $2^{Ctm} \lambda_t(d)^t = 2^{O(m)} \cdot \mathsf{poly}(d)$. Also, $\lambda_{t-1}(\log r) = (\log r)^{1+o(1)}$. Thus, the overall time complexity is bounded by

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log r).$$

□

8 The Second Algorithm over Extension Rings

The main result of this section is the following theorem.

Theorem 8.1. *Over a ring $R = (\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$, where $E(z) \in (\mathbb{Z}/r\mathbb{Z})[z]$ is a monic polynomial of degree $e \geq 1$, for all $m \in \mathbb{N}$ and sufficiently large $d \in \mathbb{N}$, there exists a deterministic algorithm that outputs the evaluation of an m-variate polynomial of degree less than d in each variable on N points in time*

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log |R|),$$

provided that $\log^{\circ c} |R| \leq d^{o(1)}$ for some fixed constant $c \in \mathbb{N}$.

Remark. *By choosing r to be a prime number p and $E(z)$ to be a monic irreducible polynomial over $\mathbb{F}_p$, we see that Theorem 1.1 follows from Theorem 8.1 assuming that the size of the finite field is at most $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of the tower of exponentials is some constant.* ◊

We present an algorithm MME-For-Extension-Rings-B, built on the algorithm MME-B in the previous section, that allows us to prove Theorem 8.1. Let $R := (\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$. Given an m-variate polynomial $f(\mathbf{x})$ over R and N evaluation points $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N \in R^m$, the algorithm outputs the evaluations of $f(\mathbf{x})$ at $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N$.

Following the algorithm in [KU11, §4.3], also presented in Section 6, our first step is lifting $f(\mathbf{x})$ to a polynomial $F(\mathbf{x}) \in (\mathbb{Z}[z])(\mathbf{x})$, and similarly lifting the evaluation points $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ to $\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \ldots, \tilde{\mathbf{a}}_N \in \mathbb{Z}[z]^m$, such that the coefficients or coordinates of the lifts are reasonably bounded. More specifically, the coefficients of $F(\mathbf{x})$ and the coordiantes of $\mathbf{a}_i$'s are polynomials in $\mathbb{Z}[z]$ with degree at most $e-1$ and the coefficients are in $[\![r]\!]$. Let $M := d^m(e(r-1))^{(d-1)m+1}+1$. Then, Kedlaya and Umans observed that for all $i \in [N]$, $F(\tilde{\mathbf{a}}_i)$ is a polynomial in z with degree at most $(e-1)dm$ and the coefficients are less than M. Therefore, if we know $F(\tilde{\mathbf{a}}_i)$ at $z = M$, we can compute $F(\tilde{\mathbf{a}}_i)$. Since the evaluation of $F(\tilde{\mathbf{a}}_i)$ at $z = M$ is less than $r' = M^{(e-1)dm+1}$, [KU11] reduces the computing of $F(\tilde{\mathbf{a}}_i)$ to the computing of $F(\mathbf{x})$ mod $(r', z - M)$. In this way, [KU11] showed that we can reduce the multivariate multipoint evaluation problem over R to that over $\mathbb{Z}[z]/(r', z - M) = \mathbb{Z}/r'\mathbb{Z}$, and the latter can be solved by the algorithm MME-B (Algorithm 4).

The problem with applying this reduction to prove Theorem 8.1 is that r' is too large for us, being exponential in m^2. If we simply use this reduction, the resulting algorithm would have the claimed time complexity only when m is at most $(\exp(\exp(\exp(\cdots(\exp(d))))))$, where the height of the tower of exponentials is some constant.

To resolve this issue, we give a more efficient reduction, which leads to the algorithm MME-For-Extension-Rings-B in this section. The following is a brief overview of the algorithm.

Overview. Our goal is to compute $F(\tilde{\mathbf{a}}_i)$ for each $i \in [N]$, whose remainder modulo $(r, E(z))$ would give the desired evaluation $f(\mathbf{a}_i)$. As mentioned above, for each $i \in [N]$, $F(\tilde{\mathbf{a}}_i)$ is a polynomial in $\mathbb{Z}[z]$ with degree less than $(e-1)md + 1$ and the coefficients are less than M. In [KU11], they interpolated $F(\tilde{\mathbf{a}}_i)$ from its evaluation at a single point $z = M$. Contrary to them, we interpolate $F(\tilde{\mathbf{a}}_i)$ from its evaluation at multiple points. Since the degree of $F(\mathbf{a}_i)$ is less than $(e-1)dm + 1$, one way can be to compute the evaluations of $F(\mathbf{a}_i)$ at all the points in $[\![(e-1)dm + 1]\!]$, and then interpolate $F(\mathbf{a}_i)$. This would reduce the computation of $F(\mathbf{a}_i)$ for all $i \in [N]$ to $(e-1)dm + 1$ instances of multivariate multipoint evaluation over $\mathbb{Z}$. Next, to solve those instances using the algorithm MME-B (Algorithm 4), we need to reduce them as instances of multivariate multipoint evaluation over rings of form $\mathbb{Z}/r'^s\mathbb{Z}$ where $s \in [m]$. For proving Theorem 8.1, we want the value of r' is independent of m. Since the coefficients of $F(\mathbf{a}_i)$ are less than M, it would be sufficient for us to compute $F(\mathbf{a}_i)$ mod M. Since $M^{1/m} = O(d(er)^d)$, which is independent of m, one can pick r' as an integer greater than $M^{1/m}$ and try to work over the ring $\mathbb{Z}/r'^m\mathbb{Z}$. However, there is a problem with this approach. To interpolate $F(\tilde{\mathbf{a}}_i)$ mod r'^m from its evaluations at the points in $[\![(e-1)dm + 1]\!]$, we need to

ensure that $(j-j')$ is a unit in the ring $\mathbb{Z}/r'^m\mathbb{Z}$ for distinct $j, j' \in [\![(e-1)dm+1]\!]$. One possible way to satisfy both the constraints is by picking r' as a prime power greater than $M^{1/m}$ for some prime larger than $(e-1)dm$. This imposes that r' has to be at least $(e-1)dm$, which makes it dependent on m. To overcome this issue, we do the following.

1. Let $\ell = ed$ and P be a prime in $[\ell, 2\ell]$. Then, we pick r' as a P-power greater than $M^{1/m}$.
2. For every $j \in [\![\ell]\!]$, we compute $F(\tilde{\mathbf{a}}_i) \bmod (r'^m, (z-j)^m)$.

The first condition ensures that r' is independent of m and for distinct $j, j' \in [\![\ell]\!]$, $(j-j')$ is a unit in $\mathbb{Z}/r'^m\mathbb{Z}$. Therefore, from $F(\mathbf{a}_i) \bmod (r'^m, (z-j)^m)$ for all $j \in [\![\ell]\!]$, we get $F(\tilde{\mathbf{a}}_i)$ modulo r'^m using Hermite interpolation (Lemma 3.9) over $\mathbb{Z}/r'^m\mathbb{Z}$. This is sufficient to compute $F(\tilde{\mathbf{a}}_i)$, since the coefficients of $F(\tilde{\mathbf{a}}_i)$ are less than r'^m.

How do we compute $F(\tilde{\mathbf{a}}_i) \bmod (r'^m, (z-j)^m)$? Consider the Taylor expansion $\tilde{\mathbf{a}}_i \equiv \sum_{u=0}^{e-1} \mathbf{a}_{i,j,u}(z-j)^u \bmod r'^m$ where the coefficients $\mathbf{a}_{i,u} \in \mathbb{Z}/r'^m\mathbb{Z}$. Let $F(\mathbf{x}) \equiv \sum_{u=0}^{e-1} f_u(\mathbf{x}) z^u \bmod r'^m$ where $f_u(\mathbf{x}) \in (\mathbb{Z}/r'^m\mathbb{Z})[\mathbf{x}]$ is the coefficient of z^u in $F(\mathbf{x}) \bmod r'^m$. This implies that $F(\tilde{\mathbf{a}}_i) \bmod (r'^m, (z-j)^m)$ is same as the $\sum_{i=1}^{e-1}(f_u(\tilde{\mathbf{a}}_i) z^u \bmod (z-j)^m)$ since the choice of r' ensures that $\tilde{\mathbf{a}}_i \bmod r'^m$ is same as $\tilde{\mathbf{a}}_i$. From Lemma 7.2, $f_u(\tilde{\mathbf{a}}_i) z^u \bmod (z-j)^m$ can be computed from the evaluations of $f_{\mathbf{e},u}(\mathbf{x})$ at $\mathbf{a}_{i,j,0}$ for all $\mathbf{e} \in \mathbb{N}^n$ with $|\mathbf{e}|_1 < m$. This shows that the evaluation of $F(\tilde{\mathbf{a}}_i) \bmod (r'^m, (z-j)^m)$ can be computed from the evaluations of $f_{\mathbf{e},u}(\mathbf{x})$ at $\mathbf{a}_{i,j,0}$ for all $\mathbf{e} \in \mathbb{N}^n$ with $|\mathbf{e}|_1 < m$. Thus, for every $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < m$, $j \in [\![\ell]\!]$ and $u \in [\![e]\!]$, we need to solve the following instance of multivariate multipoint evaluation $(f_{\mathbf{e},u}, \mathbf{a}_{1,j,0}, \mathbf{a}_{2,j,0}, \ldots, \mathbf{a}_{N,j,0})$ over the ring $\mathbb{Z}/r'^m\mathbb{Z}$. The latter problem can be solved using the algorithm MME-B (Algorithm 4).

8.1 The Description of the Algorithm

Now, we formally describe the algorithm MME-For-Extension-Rings-B.

Algorithm 5 The Second Algorithm over Extension Rings

Algorithm MME-For-Extension-Rings-B$(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, t, E(z))$

where $f(\mathbf{x})$ is an m-variate polynomial over $R := (\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ of individual degree at most $d-1$, $E(z) \in (\mathbb{Z}/r\mathbb{Z})[z]$ is a monic polynomial of degree $e \geq 1$, $\mathbf{a}_1, \mathbf{a}_2 \ldots, \mathbf{a}_N$ are evaluation points in R^m, and $t \geq 0$ is the depth of the reduction tree.

1. Compute $F(\mathbf{x}) \in (\mathbb{Z}[z])[\mathbf{x}]$ as a lift of $f(\mathbf{x})$ such that every coefficient of $F(\mathbf{x})$ is a polynomial in z of degree at most $e-1$ with coefficients in $[\![r]\!]$.

2. For all $i \in [N]$, compute $\tilde{\mathbf{a}}_i \in \mathbb{Z}[z]^m$ as a lift of $\mathbf{a}_i$ such that every coordinate of $\tilde{\mathbf{a}}_i$ is a polynomial in z of degree at most $e-1$ with coefficients in $[\![r]\!]$.

3. Let $\ell = ed$. Choose a prime $P \in [\ell, 2\ell]$. Choose the smallest P-power r' such that $r' \geq d(er)^d$. Compute $\bar{F}(\mathbf{x}) := F(\mathbf{x}) \bmod r'^m \in \mathbb{Z}/r'^m\mathbb{Z}$. For $i \in [N]$, compute $\bar{\mathbf{a}}_i := \tilde{\mathbf{a}}_i \bmod r'^m \in (\mathbb{Z}/r'^m\mathbb{Z})[z]^m$.

4. Compute $f_0(\mathbf{x}), f_1(\mathbf{x}) \ldots, f_{e-1}(\mathbf{x}) \in (\mathbb{Z}/r'^m\mathbb{Z})[\mathbf{x}]$ such that $\bar{F}(\mathbf{x}) = \sum_{u=0}^{e-1} f_u(\mathbf{x}) z^u$. For all $i \in [N]$ and $j \in [\![\ell]\!]$, compute $\mathbf{a}_{i,j,0}, \mathbf{a}_{i,j,1}, \ldots, \mathbf{a}_{i,j,e-1} \in (\mathbb{Z}/r'^m\mathbb{Z})^m$ such that $\bar{\mathbf{a}}_i = \sum_{u=0}^{e-1} \mathbf{a}_{i,j,u}(z-j)^u$.

5. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < m$ and $u \in [\![e]\!]$, use Lemma 3.6 to compute $f_{\mathbf{e},u}(\mathbf{x}) := \overline{\partial}_{\mathbf{e}}(f_u)(\mathbf{x}) \in (\mathbb{Z}/r'^m\mathbb{Z})[\mathbf{x}]$.

6. For all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < m$, $j \in [\![\ell]\!]$, and $u \in [\![e]\!]$, invoke MME-B with input $(f_{\mathbf{e},u}, \mathbf{a}_{1,j,0}, \mathbf{a}_{2,j,0}, \ldots, \mathbf{a}_{N,j,0}, r', m, t)$ to compute $f_{\mathbf{e},u}(\mathbf{a}_{i,j,0}) \in \mathbb{Z}/r'^m\mathbb{Z}$ for all $i \in [N]$.

7. For all $i \in [N]$ and $j \in [\![\ell]\!]$, compute

$$b_{i,j}(z) = \left(\sum_{u=0}^{e-1} \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < m} f_{\mathbf{e},u}(\mathbf{a}_{i,j,0}) (\bar{\mathbf{a}}_i - \mathbf{a}_{i,j,0})^{\mathbf{e}} z^u \right) \bmod (z-j)^m$$

   which is a polynomial of degree at most $m-1$ in z over $\mathbb{Z}/r'^m\mathbb{Z}$.

8. For all $i \in [N]$, use Hermite interpolation (Lemma 3.9) to compute the unique polynomial $b_i(z) \in (\mathbb{Z}/r'^m\mathbb{Z})[z]$ of degree less than ℓm such that $b_i(z) \equiv b_{i,j}(z) \pmod{(z-j)^m}$ for $j \in [\![\ell]\!]$.

9. For all $i \in [N]$, lift $b_i(z)$ to $B_i(z) \in \mathbb{Z}[z]$ with coefficients in $[\![r'^m]\!]$, and output the remainder of $B_i(z)$ modulo $(r, E(z))$ as $f(\mathbf{a}_i)$.

8.2 The Correctness of Algorithm 5

We prove the correctness of the algorithm MME-For-Extension-Rings-B (Algorithm 5), as stated by the following claim.

Claim 8.2. *Given the input* $(f(\mathbf{x}), \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, t, E(z))$*, the algorithm* MME-For-Extension-Rings-B *computes* $f(\mathbf{a}_i)$ *for all* $i \in [N]$.

Proof. The first two steps of the algorithm compute the lifts $F(\mathbf{x})$ and $\tilde{\mathbf{a}}_i$ for $i \in [N]$. As shown

in Section 6.2, for each $i \in [N]$, the degree of $F(\tilde{\mathbf{a}}_i) \in \mathbb{Z}[z]$ is bounded by $(e-1)md$, and the coefficients of $F(\tilde{\mathbf{a}}_i)$ are non-negative integers less than $M := d^m(e(r-1))^{(d-1)m+1} + 1$.

Next, we let $\ell = ed$ and compute a prime $P \in [\ell, 2\ell]$, which can be done by Bertrand's postulate. Then we find the smallest P-power r' such that $r' \geq d(er)^d$, which guarantees $r'^m \geq d^m(er)^{dm} \geq M$. Next, we compute $\bar{F}(\mathbf{x}) := F(\mathbf{x}) \bmod r'^m$ and for $i \in [N]$, compute $\bar{\mathbf{a}}_i := \tilde{\mathbf{a}}_i \bmod r'^m \in (\mathbb{Z}/r'^m\mathbb{Z})[z]^m$, so that $\bar{F}(\bar{\mathbf{a}}_i) = F(\tilde{\mathbf{a}}_i) \bmod r'^m$. As the coefficients of $F(\tilde{\mathbf{a}}_i)$ are non-negative integers less than $M \leq r'^m$, to compute $F(\tilde{\mathbf{a}}_i)$, we just need to first compute $\bar{F}(\bar{\mathbf{a}}_i)$ and then lift its coefficients to integers in $[\![r'^m]\!]$. This is precisely what the remaining steps (Steps 4–9) do.

In Step 4, we compute the data $f_u(\mathbf{x}) \in (\mathbb{Z}/r'^m\mathbb{Z})[\mathbf{x}]$ and $\mathbf{a}_{i,j,u} \in (\mathbb{Z}/r'^m\mathbb{Z})^m$ such that $\bar{F}(\mathbf{x}) = \sum_{u=0}^{e-1} f_u(\mathbf{x}) z^u$ and $\bar{\mathbf{a}}_i = \sum_{u=0}^{e-1} \mathbf{a}_{i,j,u}(z-j)^u$. This is possible as the coefficients of $\bar{F}(\mathbf{x})$ and the coordinates of each $\bar{\mathbf{a}}_i$ are polynomials of degree at most $e-1$ in z over $\mathbb{Z}/r'^m\mathbb{Z}$. Then in Step 5, we compute the data $f_{\mathbf{e},u}(\mathbf{x}) := \overline{\partial}_{\mathbf{e}}(f_u)(\mathbf{x}) \in (\mathbb{Z}/r'^m\mathbb{Z})[\mathbf{x}]$ for all $\mathbf{e} \in \mathbb{N}^m$ with $|\mathbf{e}|_1 < m$. And in Step 6, we compute the evaluations $f_{\mathbf{e},u}(\mathbf{a}_{i,j,0}) \in \mathbb{Z}/r'^m\mathbb{Z}$ using the algorithm MME-B.

Consider $i \in [N]$ and $j \in [\![\ell]\!]$. For $u \in [\![e]\!]$, as $\bar{\mathbf{a}}_i - \mathbf{a}_{i,j,0} = \sum_{u=1}^{e-1} \mathbf{a}_{i,j,u}(z-j)^u$ is a multiple of $z-j$, Lemma 7.2 gives

$$f_u(\bar{\mathbf{a}}_i) \equiv \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < m} f_{\mathbf{e},u}(\mathbf{a}_{i,j,0})(\bar{\mathbf{a}}_i - \mathbf{a}_{i,j,0})^{\mathbf{e}} \pmod{(z-j)^m}.$$

Therefore,

$$\bar{F}(\bar{\mathbf{a}}_i) = \sum_{u=0}^{e-1} f_u(\bar{\mathbf{a}}_i) z^u \equiv \sum_{u=0}^{e-1} \sum_{\mathbf{e} \in \mathbb{N}^m : |\mathbf{e}|_1 < m} f_{\mathbf{e},u}(\mathbf{a}_{i,j,0})(\bar{\mathbf{a}}_i - \mathbf{a}_{i,j,0})^{\mathbf{e}} z^u \pmod{(z-j)^m}.$$

We compute $b_{i,j}(z) := \bar{F}(\bar{\mathbf{a}}_i) \bmod (z-j)^m$ in Step 7 using the above equation.

In Step 8, we compute $b_i(z) = \bar{F}(\bar{\mathbf{a}}_i)$ from its remainders $b_{i,j}$ modulo $(z-j)^m$ for $i \in [N]$ and $j \in [\![\ell]\!]$ using Hermite interpolation (Lemma 3.9). As $\ell = ed$, we have $\deg_z(\bar{F}(\bar{\mathbf{a}}_i)) \leq (e-1)md < \ell m$. And as r' is a power of a prime P and $P \geq \ell$, the difference $j - j'$ has a multiplicative inverse in $\mathbb{Z}/r'^m\mathbb{Z}$ for distinct $j, j' \in [\![\ell]\!]$. So $b_i(z)$ can indeed be found using Hermite interpolation.

Finally, in Step 9, we compute the lift $F(\tilde{\mathbf{a}}_i)$ from $\bar{F}(\bar{\mathbf{a}}_i)$, and then output $f(\mathbf{a}_i) = F(\tilde{\mathbf{a}}_i) \bmod (r, E(z))$, as desired. □

8.3 The Time Complexity of Algorithm 5

We now analyze the time complexity of Algorithm 5. Step 1 takes time $O(d^m) \cdot \mathsf{poly}(m, d, \log|R|)$. And Step 2 takes time $O(N) \cdot \mathsf{poly}(m, \log|R|)$. Note $\ell, \log r' \leq \mathsf{poly}(d, \log|R|)$. Then Step 3 and Step 4 both take time $O(d^m + N) \cdot \mathsf{poly}(m, d, \log|R|)$. By Lemma 3.6, Step 5 takes time $O(\binom{2m-1}{m-1} d^m) \cdot \mathsf{poly}(m, d, \log|R|)$. Step 7 takes time $O(\binom{2m-1}{m-1} N) \cdot \mathsf{poly}(m, d, \log|R|)$. By Lemma 3.9, Step 8 takes time $O(N) \cdot \mathsf{poly}(m, d, \log|R|)$. And Step 9 takes time $O(N) \cdot \mathsf{poly}(m, \log|R|)$.

Finally, by Theorem 7.10, the time complexity of Step 6 is bounded by

$$O\left(2^{C(t+1)m} \lambda_t(d)^t \lambda_{t-1}(\log r') \left((\lambda_t(d) \log^{\circ t} r')^m + N\right)\right) \cdot \mathsf{poly}(m, d, \log|R|).$$

for $0 \leq t < \log^\star(r')$, where $r' \in [d(er)^d, 2ed^2(er)^d]$ is as in the algorithm and $C > 0$ is a large enough absolute constant. This step dominates the time complexity of the whole algorithm.

Combining the above time complexity analysis with Claim 8.2 yields the following theorem.

Theorem 8.3. *Let $f(\mathbf{x})$ be an m-variate polynomial over $R = (\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$ of individual degree less than d, where $E(z)$ is a monic polynomial of degree $e \geq 1$. Let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$ be N points in R^m. Let t be a non-negative integer less than $\log^\star(d(er)^d)$. Then, given*

$(f, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N, r, t, E(z))$, *the algorithm* MME-For-Extension-Rings-B *computes* $f(\mathbf{a}_i)$ *for all* $i \in [N]$ *in time*

$$O\left(2^{C(t+1)m}\lambda_t(d)^t\lambda_{t-1}(\log r')\left((\lambda_t(d)\log^{\circ t} r')^m + N\right)\right) \cdot \mathsf{poly}(m, d, \log|R|).$$

where $r' = 2ed^2(er)^d$ *and* $C > 0$ *is a large enough absolute constant.*

Now we are ready to prove Theorem 8.1. For convenience, we restate it here.

Theorem 8.1. *Over a ring* $R = (\mathbb{Z}/r\mathbb{Z})[z]/(E(z))$*, where* $E(z) \in (\mathbb{Z}/r\mathbb{Z})[z]$ *is a monic polynomial of degree* $e \geq 1$*, for all* $m \in \mathbb{N}$ *and sufficiently large* $d \in \mathbb{N}$*, there exists a deterministic algorithm that outputs the evaluation of an* m*-variate polynomial of degree less than* d *in each variable on* N *points in time*

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log|R|),$$

provided that $\log^{\circ c}|R| \leq d^{o(1)}$ *for some fixed constant* $c \in \mathbb{N}$.

Proof. We invoke the algorithm MME-For-Extension-Rings-B for $t = \max\{c, 2\}$, which is less than $\log^\star(d(er)^d)$ as d is sufficiently large. Then, from Theorem 8.3, we get the evaluations in time

$$O\left(2^{C(t+1)m}\lambda_t(d)^t\lambda_{t-1}(\log r')\left((\lambda_t(d)\log^{\circ t} r')^m + N\right)\right) \cdot \mathsf{poly}(m, d, \log|R|).$$

where $r' = 2ed^2(er)^d$ and $C > 0$ is a large enough absolute constant. From the definition, $\lambda_t(d) = d^{1+o(1)}$. Noting that $\log\log r' = O(\log d + \log\log|R|)$, $\log^{\circ c}|R| \leq d^{o(1)}$, and $t = \max\{c, 2\}$, we have $\log^{\circ t} r' = d^{o(1)}$. Therefore, $\lambda_t(d)\log^{\circ t} r' = d^{1+o(1)}$. Since t is bounded by some constant, $2^{C(t+1)m}\lambda_t(d)^t = 2^{O(m)} \cdot \mathsf{poly}(d)$. Also, $\lambda_{t-1}(\log r') \leq \mathsf{poly}(d, \log|R|)$. Thus, the overall time complexity is bounded by

$$(d^m + N)^{1+o(1)} \cdot \mathsf{poly}(m, d, \log|R|).$$

□

Acknowledgement

Mrinal is thankful to Swastik Kopparty for introducing him to the question of multipoint evaluation and the work of Kedlaya–Umans [KU11] and to Prahladh Harsha and Ramprasad Saptharishi for many helpful discussions.

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[BGKM21] Vishwas Bhargava, Sumanta Ghosh, Mrinal Kumar, and Chandra Kanta Mohapatra. Fast, Algebraic Multivariate Multipoint Evaluation in Small Characteristic and Applications. *arXiv preprint arXiv:2111.07572*, 2021. To appear in STOC 2022.

[BKW19] Andreas Björklund, Petteri Kaski, and Ryan Williams. Generalized Kakeya Sets for Polynomial Evaluation and Faster Computation of Fermionants. *Algorithmica*, 81(10):4010–4028, 2019.

[BM74] Allan Borodin and Robert Moenck. Fast Modular Transforms. *Journal of Computer and System Sciences*, 8(3):366–386, 1974.

[Bom65] Enrico Bombieri. On the Large Sieve. *Mathematika*, 12:201–225, 1965.

[For14] Michael Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, Massachusetts Institute of Technology, 2014.

[Juk11] Stasys Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer, second edition, 2011.

[Ked15] Kiran Kedlaya. Lecture notes for the course 'Analytic Number Theory', 2015.

[KU11] Kiran Kedlaya and Christopher Umans. Fast Polynomial Factorization and Modular Composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011.

[Lan02] Serge Lang. *Algebra*. Springer-Verlag, New York Inc., third edition, 2002.

[May20] James Maynard. Primes in Arithmetic Progressions to Large Moduli I: Fixed Residue Classes. *arXiv preprint arXiv:2006.06572*, 2020.

[NZ04] Michael Nüsken and Martin Ziegler. Fast Multipoint Evaluation of Bivariate Polynomials. In *Algorithms – ESA 2004*, pages 544–555, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Sho08] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, New York, second edition, 2008. Available from `https://shoup.net/ntb/`.

[Sie35] Carl Siegel. Über die Classenzahl quadratischer Zahlkörper. *Acta Arithmetica*, 1(1):83–86, 1935.

[Uma08] Christopher Umans. Fast Polynomial Factorization and Modular Composition in Small Characteristic. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 481–490. ACM, 2008.

[Vin65] A. I. Vinogradov. The Density Hypothesis for the Dirichlet L-series. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 29:903–934, 1965.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, third edition, 2013.

[Wal36] Arnold Walfisz. Zur additiven Zahlentheorie. II. *Mathematische Zeitschrift*, 40(1):592–607, 1936.