Counting basic-irreducible factors mod p^k in deterministic poly-time and p-adic applications

Ashish Dwivedi * Rajat Mittal † Nitin Saxena ‡

Abstract

Finding an irreducible factor, of a polynomial $f(x)$ modulo a prime p, is not known to be in deterministic polynomial time. Though there is such a classical algorithm that *counts* the number of irreducible factors of $f \bmod p$. We can ask the same question modulo prime-powers p^k. The irreducible factors of $f \bmod p^k$ blow up exponentially in number; making it hard to describe them. Can we count those irreducible factors mod p^k that remain irreducible mod p? These are called *basic-irreducible*. A simple example is in $f = x^2 + px \bmod p^2$; it has p many basic-irreducible factors. Also note that, $x^2 + p \bmod p^2$ is irreducible but not basic-irreducible!

We give an algorithm to count the number of basic-irreducible factors of $f \bmod p^k$ in deterministic $\mathrm{poly}(\deg(f), k \log p)$-time. This solves the open questions posed in (Cheng et al, ANTS'18 & Kopp et al, Math.Comp.'19). In particular, we are counting roots mod p^k; which gives the first deterministic poly-time algorithm to compute Igusa zeta function of f. Also, our algorithm efficiently partitions the set of all basic-irreducible factors (possibly exponential) into merely $\deg(f)$-many disjoint sets, using a compact tree data structure and *split* ideals.

2012 ACM CCS concept: Theory of computation– Algebraic complexity theory, Pseudorandomness and derandomization; Computing methodologies– Algebraic/ Number theory algorithms, Hybrid symbolic-numeric methods; Mathematics of computing– Combinatoric problems.
Keywords: deterministic, root, counting, modulo, prime-power, tree, basic irreducible, unramified.

1 Introduction

Factoring a univariate polynomial, over *prime* characteristic, is a highly well studied problem. Though efficient factoring has been achieved using randomization, still efficient derandomization is a longstanding problem. A related question of equal importance is root finding, but this is known to be equivalent to factoring in deterministic poly-time. Surprisingly, testing irreducibility, or even counting irreducible factors, is easy in this regime. The main tool here is the magical Frobenius morphism of prime p characteristic rings: $x \mapsto x^p$.

Though much effort has been put in prime characteristic, few results are known in *composite* characteristic n [Sha93]. Even irreducibility testing of a polynomial, with the prime factorization of n given, has no efficient algorithm known. This reduces to *prime-power* characteristic p^k [vzGH98]. Deterministic factoring in such a ring is a much harder question (at least it subsumes deterministic factoring mod p). In fact, even randomized algorithms, or practical solutions, are currently elusive [vzGH96, vzGH98, Kli97, Săl05, Sir17, DMS19]. The main obstruction is non-unique factorization.

*CSE, Indian Institute of Technology, Kanpur, ashish@cse.iitk.ac.in
†CSE, Indian Institute of Technology, Kanpur, rmittal@cse.iitk.ac.in
‡CSE, Indian Institute of Technology, Kanpur, nitin@cse.iitk.ac.in

Being a non-unique factorization domain, there could be exponential number of roots, or irreducible factors, modulo prime-powers [vzGH96]. So one could ask a related question about counting all the irreducible factors (respectively roots) modulo prime-powers. Efficiently solving this counting problem will give us an efficient irreducibility testing criteria, which is the first question one wants to try. Recall that prime characteristic allows such an efficient method.

Motivated by this, we ask— Could we describe all factors which remain irreducible mod p? Such factors are called *basic*-irreducible in the literature. This is much more than counting roots mod p^k (as, $f(\alpha) = 0$ iff $x - \alpha$ is a basic-irreducible factor of f). These roots, besides being naturally interesting, have various applications in— factoring [Chi87, Chi94, CG00], coding theory [BLQ13, Săl05], elliptic curve cryptography [Lau04], arithmetic algebraic-geometry [ZG03, DH01, Den91, Igu74]. Towards this we design a machinery, yielding the following result:

Given a degree d integral polynomial $f(x)$ and a prime-power p^k, we partition the set of all basic-irreducible factors of $f \bmod p^k$ into at most d (compactly provided) subsets in deterministic poly$(d, k \log p)$-time; in the same time we count the number of factors in each of these subsets.

Also, we can compactly partition (and count) the roots of $f \bmod p^k$ in deterministic poly-time.

This efficient partitioning of (possibly exponentially many) roots into merely d subsets is reminiscent of the age-old fact: there are at most $\deg(g)$ roots of a polynomial $g(x)$ over a field. Root sets mod p^k are curious objects; not every subset of $\mathbb{Z}/p^k\mathbb{Z}$ is a root set (except when $k = 1$). Their combinatorial properties have been studied extensively [Sie55, CP56, Bha97, DM97, Mau01]. In this regard, our result is one more step to understand the hidden properties of root-sets mod prime-powers.

Factoring mod p^k has applications in factoring over *local* fields [Chi87, Chi94, CG00]. Previously, the latter was achieved through randomized factoring mod p [CZ81] and going to extensions of $\mathbb{Q}_p$. Directly factoring mod p^k, for arbitrary k, would imply a new and more natural factoring algorithm over p-adic fields. In fact, *our method gives the first deterministic poly-time algorithm to count basic-irreducible factors of $f \in \mathbb{Q}_p[x]$*; by picking k such that $p^{k/2} \nmid \text{discriminant}(f)$ (see [vzGH98, Thm. 3.11]). This derandomization was not known before, though $\mathbb{Q}_p[x]$ is indeed a unique factorization domain.

1.1 Previously known results

The questions of root finding and root counting of $f \bmod p^k$ are of classical interest, see [NZM13, Apo13]. Using Hensel lifting (Section A) we know how to 'lift' a root, of multiplicity one, of f mod p to a root of f mod p^k, in a unique way. But this method breaks down when the root (mod p) has multiplicity more than one. [BLQ13, Cor.4] was the first work to give an efficient randomized algorithm to count, as well as find, all the roots of f mod p^k. [NRS17] improved the time complexity of [BLQ13]. In this line of progress, very recently [CGRW18] gave a deterministic algorithm to count roots in time *exponential* in the parameter k. Extending the idea of [CGRW18], [KRRZ19] gave another efficient randomized algorithm to count roots of f mod p^k. Note that finding the roots deterministically seems a difficult problem because it requires efficient deterministic factoring of f mod p (which is a classical open problem). But counting the roots mod p^k deterministically may be an easier first step.

Recently there has been some progress in factoring f mod p^k when k is constant. [DMS19] gave the first efficient randomized algorithm to factor $f \bmod p^k$ for $k \leq 4$. This gives an exponential improvement over the previous best algorithms of [Sir17, vzGH98, vzGH96] mod p^k ($k \leq 4$). In

fact, they generalized Hensel lifting method to mod p^k, for $k \leq 4$, in the difficult case when f mod p is power of an irreducible. The related derandomization questions are all open.

The case of factoring f mod p^k when k is "large"— larger than the maximum power of p dividing the discriminant of the integral f —has an efficient randomized algorithm due to [vzGH98]. They showed, assuming large k, that factorization mod p^k is well behaved and corresponds to the unique p-adic factorization of f (i.e. in $\mathbb{Q}_p[x]$). In turn, p-adic factoring has known efficient randomized algorithms [Chi87, Chi94, CG00]. The derandomization questions are all open.

We now give a deterministic method to count all the roots (resp. basic-irreducible factors) efficiently. In fact, our proof can be seen as a deterministic poly-time reduction of basic-irreducible factor finding mod p^k to root finding mod p. In particular, it subsumes all the results of [BLQ13].

1.2 Our results

Theorem 1 (Root count)**.** *Let p be a prime, $k \in \mathbb{N}$ and $f(x) \in \mathbb{Z}[x]$. Then, all the roots of f* mod p^k *can be counted in deterministic poly($\deg f, k \log p$)-time.*

This is the first efficient derandomization of the randomized root counting algorithms [BLQ13, KRRZ19], and an exponential improvement over the recent deterministic algorithm of [CGRW18]. The challenge arises from the fact that we need to count the possibly exponentially many roots without being able to find them[1].

Remarks. 1) In the algorithm, the (possibly exponential) root-set of f mod p^k gets partitioned into at most $\deg(f)$-many disjoint subsets and we output a compact representation, called *split ideal*, for each of these subsets. We do count them, but do not yet know how to find a root deterministically.

2) This gives an efficient way to deterministically compute the Igusa zeta function, given an integral univariate f and a prime p. This follows from the fact that we just need to compute $N_k(f) :=$number of roots of f mod p^k, for $k \in [\ell]$ s.t. $p^{\ell/2} \nmid$ discriminant(f), to estimate *Poincaré series* $\sum_{i=0}^{\infty} N_i(f)x^i$ [Den91, Igu74]. Interestingly, it converges to a rational function!

The proof follows from the fact that for each $i > l$, $N_i(f)$ is the sum of $t \leq d$ many p-powers where t is constant for each $i > l$ (see [vzGH98, Thm. 3.11]). So the sum $\sum_{i>l}^{\infty} N_i(f)x^i$ converges.

3) This is the first deterministic poly-time algorithm to count the number of lifts of a *repeated* root of f mod p to f mod p^k.

4) This gives the first deterministic poly-time algorithm to count the number of p-adic integral roots of a given p-adic polynomial $f \in \mathbb{Q}_p[x]$. (Count roots mod p^ℓ where $p^{\ell/2} \nmid$ discriminant(f) [vzGH98, Thm. 3.11].)

Next, we extend the ideas for counting roots to count all the basic-irreducible factors of f mod p^k in deterministic polynomial time. Recall that a *basic-irreducible* factor of f mod p^k is one that remains irreducible in mod p arithmetic.

Theorem 2 (Factor count)**.** *Let p be a prime, $k \in \mathbb{N}$ and $f(x) \in \mathbb{Z}[x]$. Then, all the basic-irreducible factors of f* mod p^k *can be counted in deterministic poly($\deg f, k \log p$)-time.*

We achieve this by extending the idea of counting roots to more general p-adic integers. Essentially, we efficiently count all the roots of $f(x)$ in $\mathcal{O}_K/\langle p^k \rangle$, where $\mathcal{O}_K$ is the ring of integers of a p-adic *unramified* extension $K/\mathbb{Q}_p$ (refer [Kob77] for the standard notation). Currently, there is no fast, practical method known to find/count roots when K is *ramified*.

[1]Note that counting roots of a multivariate polynomial over a finite field is #P-complete, even if the degree is restricted to be three [EK90].

Corollary 3. *Consider (an unknown) p-adic extension $K := \mathbb{Q}_p[y]/\langle g(y)\rangle$, which is unramified and has degree Δ. Let $f(x) \in \mathbb{Z}[x]$, p, k, Δ be given as input (in binary).*

Then, we can count all the roots of f, in $\mathcal{O}_K/\langle p^k\rangle$, in deterministic poly($\deg(f), k \log p, \Delta$)-time.

Remarks. 1) This gives the first deterministic poly-time algorithm to count the number of (unramified p-adic integral) roots of a given p-adic polynomial $f \in K[x]$.

2) Our method generalizes to efficiently count all the roots of a given polynomial $f(x) \in (\mathbb{F}[t]/\langle h(t)^k\rangle)[x]$ for a given polynomial h (resp. $f \in \mathbb{F}[[t]][x]$ with power-series coefficients); assuming that $\mathbb{F}$ is a field over which root counting is efficient (eg. $\mathbb{Q}, \mathbb{R}, \mathbb{F}_p$ and their algebraic extensions).

1.3 Proof techniques

Our implementation involves constructing a *list data structure* $\mathcal{L}$ which implicitly partitions the root-set of $f \bmod p^k$ into at most $\deg(f)$-many disjoint subsets; and count the number of roots in each such subset. The construction of $\mathcal{L}$ is incremental, by doing arithmetic modulo special ideals,

Split ideals. A *split ideal* I_l of length $l+1$, and degree b, is a 'triangular' ideal defined as $I_l = \langle h_0(x_0), h_1(\bar{x}_1), \ldots, h_l(\bar{x}_l)\rangle$, where the notation $\bar{x}_i$ refers to the variable set $\{x_0, \ldots, x_i\}$ and $b = \prod_{0\le i\le l} \deg_{x_i}(h_i)$. It implicitly stores a size-b subset of the root-set of $f \bmod p^k$, where a root looks like $\sum_{0\le i\le l} x_i p^i$ till precision p^{l+1}. Note that a root r of $f \bmod p^k$ is also a root of $f \bmod p^l$ for all $l \in [k]$. Since we cannot access them directly, we 'virtualize' them in the notation $\bar{x}_l$.

The structure of these ideals is quite nice and recursive (Section 2). So it may keep splitting (in Algorithm 1) till it becomes a *maximal ideal*, which corresponds to a single point in $(\mathbb{F}_p)^l$ and has degree one. Or, the algorithm may halt earlier, due to 'stable clustering' of roots, and then we call the ideals– *maximal split ideal*; in fact, $\mathcal{L}$ has only maximal split ideals. These do not give us the actual roots but do give us their count!

List data structure. $\mathcal{L}$ implicitly stores, and may partition, the root-set of $f \bmod p^k$. Essentially, $\mathcal{L}$ is a set of at most d maximal split ideals, i.e. $\mathcal{L} = \{I_1(l_1, d_1), \ldots, I_n(l_n, d_n)\ \}$, where each ideal $I_j \subseteq \mathbb{F}_p[\bar{x}_{k-1}]$ has two parameters— length l_j and degree d_j. A maximal split ideal $I(l, D)$ implicitly stores a size-D subset of the root-set of $f \bmod p^k$. This yields a simple count of Dp^{k-l} for the corresponding roots. Ideals in $\mathcal{L}$ have the property that they represent disjoint subsets of roots; and they collectively represent the whole root-set of $f \bmod p^k$. Thus, $\mathcal{L}$ gives us both the (implicit) structure and the (exact) size of the root-set of $f \bmod p^k$. In the intermediate steps of the algorithm, for efficiency reasons, we will store a tuple (I_j, f_{I_j}) in a changing *stack* S. Where, $f_{I_j}(\bar{x}_{l_j-1}, x) := f(x_0 + px_1 + \cdots + p^{l_j-1}x_{l_j-1} + p^{l_j}x) \bmod \hat{I}_j$ is a 'shifted and reduced' version of f tagging along (with x as the only *free* variable).

Roots-tree data structure. Most importantly, we need to prove that $|\mathcal{L}|$, and the degree of the split ideals in $\mathcal{L}$, remains at most $\deg(f)$ at all times in the algorithm (while $f \bmod p^k$ may have exponentially many roots). To achieve this, we use a different way to look at the data structure $\mathcal{L}$— in *tree* form RT where each generator h_i appearing in an $I \in \mathcal{L}$ appears as an edge of the tree; conversely, each tree node v denotes the intermediate split ideal corresponding to the path from the root (of the tree RT) to v.

The roots-tree RT has a useful parameter at every node– degree. Degree of a node measures the possible extensions to the next level, and it possesses the key property: it 'distributes' to its children degrees. This helps us to simultaneously bound the width of RT and degree of split ideals, to be at most the degree $\deg(f)$ of the root node. Otherwise, since we compute with k-variate polynomials, a naive analysis of the tree-size (resp. degree of split ideals) would give a bound of $\deg(f)^k$, or a slightly better $\deg(f)2^k$ as in [CGRW18, pg.9]; which is exponential in the input size $\deg(f) \cdot k \log p$.

1.4 Proof overview

Proof idea of Theorem 1. Let $R := \mathbb{Z}/\langle p^k \rangle$; so $R/\langle p \rangle \cong \mathbb{F}_p$. Let $\mathcal{Z}_R(f)$ be the zeroset of $f \bmod p^k$.

The idea to count roots of $f \bmod p^k$ comes from the elementary fact: Any root $r \in R$ of $f \bmod p^k$ can be seen in a p-adic (or base-p) representation as $r =: r_0 + pr_1 + p^2 r_2 + \ldots + p^{k-1} r_{k-1}$, for each $r_i \in \mathbb{F}_p$. Thus, we decompose our formal variable x into multi-variables $x_0, \ldots, x_{k-1}$ being related as, $x = x_0 + px_1 + p^2 x_2 + \ldots + p^{k-1} x_{k-1}$.

Though, getting roots of $f(x_0) \bmod p$ deterministically is difficult, we can get the count on the number of roots of $f(x_0) \bmod p$ from the degree of a polynomial $h(x_0) \in \mathbb{F}_p[x_0]$, which is the gcd of f and *Frobenius* polynomial $x_0^p - x_0 \bmod p$. This way of implicitly representing a set of desired objects by a polynomial and using its properties (eg. degree) to get a count on the objects is widely termed as *polynomial method*.

This gives us a length-1 and degree-$\deg_{x_0}(h_0)$ split ideal $I_0 := \langle h_0(x_0) \rangle$. Since I_0 represents all roots of $f \bmod p$, we can again apply the polynomial method to incrementally build on ideal I_0 to get greater length split ideals representing roots of f with greater precision, say mod p^{l+1}.

To do this, we trivially lift I_0 to make it an ideal $\hat{I}_0$ in R. Solve $f(x_0 + px) \equiv p^\alpha g(x_0, x) \bmod \hat{I}_0$ for $\alpha \in \mathbb{N}$ and $g \not\equiv 0 \bmod p$. Reduce $g(x_0, x)$ over $\mathbb{F}_p$ again, and calculate the next set of candidates for x_1 implicitly in a polynomial $h_1 \in \mathbb{F}_p[x_0, x]$ defined as, $h_1 := \mathrm{GCD}(g(x_0, x) \bmod p, x^p - x) \bmod I_0$. Using the properties of split ideal (Lemma 11), multivariate-gcd modulo I_0 yields h_1 that 'stores' all the candidates for x_1, for each root x_0 represented by I_0. So, we get a length 2 split ideal $I_1 := I_0 + \langle h_1(x_0, x_1) \rangle$.

In every iteration, we add a new variable, by solving equations like $f(x_0 + px_1 + p^2 x_2 + \ldots + p^l x_l + p^{l+1} x) \equiv p^\alpha g(\bar{x}_l, x)$ modulo a length $l+1$ triangular ideal $\hat{I}_l$, for $\alpha \in \mathbb{N}$ and $g \not\equiv 0 \bmod p$. This gives us the next candidate $h_{l+1}(\bar{x}_l, x) := \mathrm{GCD}(g(\bar{x}_l, x) \bmod p, x^p - x) \bmod I_l$; moving to a more precise split ideal. Sometimes we get that g and $x^p - x$ are coprime mod I_l, those cases indicate *dead-end* and we stop processing those branches. Finally we reach $\alpha = k$, which indicates *full precision*; and we get a maximal split ideal I_l which we add to the list $\mathcal{L}$.

Division by 'zero'. Some computations modulo a split ideal may not be possible. These cases arise only due to *zerodivisors*. In those cases, we will exploit the zerodivisor to split/factor the current split ideal into more split ideals of smaller degree. We can keep track of all these split ideals using a stack and keep performing the same computations iteratively. Since a split ideal has finite length, the process must terminate. The real challenge lies in proving a good bound.

Efficiency via roots-tree. Now, we need to show that the algorithm to construct $\mathcal{L}$ is efficient and that $|\mathcal{L}| \leq \deg(f)$ (in fact, sum of degrees of all maximal split ideals in $\mathcal{L}$ is at most $\deg(f)$). In a particular iteration, the algorithm just performs routine computations like– reduction modulo the current split ideal I, inversion, zerodivisor testing, gcd, exponentiation, and computing p-valuations or multiplicities; which are clearly bounded by $\mathrm{poly}(\deg(f), k \log p, \deg(I))$ (Sections C & D). It is harder to bound the number of iterations and $\deg(I)$.

To understand the number of iterations, we review the construction of $\mathcal{L}$ as the formation of a tree, which we call roots-tree RT. A node of RT corresponds to an intermediate split ideal I, where an edge at level i on the path from the root (of RT) to the node corresponds to the generator $h_i(\bar{x}_i)$ of I. Each time we update a split ideal I_{l-1} to $I_l := I_{l-1} + \langle h_l \rangle$ we add a child, to the node corresponding to I_{l-1}, hanging by a new edge labelled h_l. Similarly, splitting of an ideal at some generator $h_i(\bar{x}_i)$ into m ideals corresponds to creating m subtrees hanging by edges which are m copies of the edge labelled h_i. This way the roots-tree upper bounds the number of iterations;

moreover, the maximal split ideals in $\mathcal{L}$ appear as leaves in RT.

Degree distribution in RT. Each node N of RT has an associated parameter, 'degree of node' $[N]$ (Definition 15), which is defined in such a way that it distributes to degree of its children (i.e. $[N]$ is at least the sum of degrees of its child nodes). This is intended to measure the possible extensions x_l modulo the corresponding split ideal I_{l-1}, and is a suitable multiple of $\deg(I_{l-1})$. Applying degree's property inductively, we get that the degree of root node of RT, which is $\deg(f)$, distributes to the degree of the leaves and so the sum of degrees of all maximal split ideals in $\mathcal{L}$ is at most $\deg(f)$. The distributive property of $[N]$, corresponding to ideal I_{l-1}, comes from the fact: the degree of a child C corresponding to ideal $I_l = I_{l-1} + \langle h_l \rangle$ is bounded by the *multiplicity* of roots of $h_l(\bar{a}, x)$ times $\deg(I_{l-1})$, corresponding to some root $\bar{a}$ of I_{l-1}; and the overall sum of these multiplicities for every child of N is naturally bounded by the degree of N (Lemma 16).

The details are given in Section 3.

Proof idea of Theorem 2. The idea, and even the algebra, is the same as for Theorem 1. The definition of list $\mathcal{L}$ easily extends to implicitly store all the basic-irreducible factors of $f \bmod p^k$ of some degree b (a generalization over roots which corresponds to degree $b = 1$ basic-irreducible factors). This uses a strong property possessed by basic-irreducible factors. A basic-irreducible factor $g(x) \in (\mathbb{Z}/\langle p^k \rangle)[x]$ of $f \bmod p^k$, of degree b, completely splits over the *Galois ring* $G(p^k, b) := \mathbb{Z}[y]/\langle p^k, \varphi(y) \rangle$, where $\varphi(y) \bmod p$ is an irreducible of degree b (Section A). Conversely, if we find a root of $f(x)$, in $G(p^k, b)$, then we find a degree-b basic-irreducible factor of $f \bmod p^k$.

By distinct degree factorization we can assume $f(x) \equiv (\varphi_1 \dots \varphi_m)^e + ph(x) \bmod p^k$, where each $\varphi_i(x) \bmod p$ is irreducible and degree-b. We construct $\mathcal{L}$ by applying the algorithm of Theorem 1, with one change: every time to update a length-l split ideal I_{l-1} to a length $l+1$ ideal $I_l := I_{l-1} + \langle h_l \rangle$, we compute h_l using the Frobenius polynomial $x^q - x \bmod p$, where $q := p^b$. Basically, for x, we focus on $\mathbb{F}_q$-roots instead of the erstwhile $\mathbb{F}_p$-roots.

We count the number of (distinct, monic, degree-b) basic-irreducible factors represented by each maximal split ideal $I(l, D) \in \mathcal{L}$ as: Dq^{k-l}/b. The details are given in Section 4.

1.5 Comparison with previous works

Broad outline: As already mentioned in Section 1.1, [BLQ13] was the first work to give an efficient algorithm to count, as well as find, all the roots of $f \bmod p^k$. The outline of [BLQ13] is as follows:

(1) Consider the roots of $f \bmod p^k$ in p-adic form as $r =: \Sigma_{i=0}^{k-1} r_i p^i$.

(2) Design an algorithm which builds up root r incrementally by finding candidates for the r_i's. The analysis of this gives rise to an exploration tree (which we call *roots-tree*).

(3) Bound the size of the roots-tree to show the efficiency of the algorithm.

At the most basic level, all the subsequent works, i.e. [CGRW18] and [KRRZ19], as well as the present work, follow the above outline.

Previous randomized algorithms: The crux of the algorithm of [BLQ13], is that at every iteration it reduces finding the roots of $f(x) \bmod p^k$ to $f_a(x) \bmod p^{k-\alpha}$, where $f_a(x)$ is an integral polynomial defined as $f_a(x) := f(a + px)/p^\alpha$, for every root a of $f \bmod p$ and $p^\alpha || f(a + px)$. They used known randomized algorithms over $\mathbb{F}_p$ to get each such root a.[2]

[2] We refer to [DMS19, Appdx.B] for a simple exposition of the algorithm of [BLQ13], after replacing φ by p there.

The efficiency of their algorithm was based upon the observation that the degree of a node represented by f_a in the roots-tree is at most the multiplicity of the root a. This way [BLQ13] could show that, although exponential, the roots of $f \bmod p^k$ can be divided into at most d many easily representable clusters. The randomized root counting algorithm of [KRRZ19] also follows the same multiplicity argument to show the efficiency of their algorithm ([KRRZ19, Lem.3.6]). Unlike [BLQ13], even with randomization, [KRRZ19] only do root-counting mod p^k.

Challenges in derandomization: It is challenging to extend the properties, of the randomized algorithms, to the deterministic (poly-time) context. The big questions that remained open were—

(1) Can we still cluster the roots of $f \bmod p^k$ deterministically? Note that efficient root finding algorithm over $\mathbb{F}_p$ is unavailable now in deterministic context.

(2) Can we get (may be implicitly) d clusters, deterministically?

(3) Can we generalize the multiplicity argument, first introduced in [BLQ13], in the deterministic context to show that size of the corresponding roots-tree is small?

(4) Can we extend the techniques to count basic-irreducible factors $f \bmod p^k$, deterministically?

Deterministic Algorithms: Prior to our work, [CGRW18] was the best known deterministic algorithm (taking time exponential in k) for root-counting. The work of [CGRW18] was the first to give the idea of storing roots implicitly in ideals of triangular form, which we call split ideals. In that sense, the outline of both [CGRW18] and our work are the same— start with the split ideal of length one and keep growing it into more and more split ideals until we get all the maximal split ideals.

However, our algorithm to construct all the maximal split ideals, in particular *growing* and *splitting* intermediate split ideals, are quite different from [CGRW18]. [CGRW18] relies on special Grobner basis computation, and ideal decomposition algorithms, to grow and split ideals. For example, unlike [CGRW18]'s complexity bound of $\text{poly}(\deg(f), 2^k \log p)$ to compute $f(x_0+px_1+\ldots+p^l x_l+p^{l+1}x) =: p^\alpha g(\bar{x}_l, x) \bmod I_l$, we manage to optimize the time-complexity to $\text{poly}(\deg(f), k \log p)$ deterministically.

In lieu of Grobner basis, our algorithm relies on some simple key properties of split ideals (we systematically define and emphasize the key property of split ideals in Section 2). We also develop the algebra to perform computation (such as reduction, gcd, and testing for zero divisors) modulo a split ideal efficiently. Given these properties and subroutines, growing a split ideal $I_l = \langle h_0(x_0), \ldots, h_l(\bar{x}_l)\rangle$ is just a gcd computation modulo I_l (see the proof idea Section 1.4). Whenever we have a factorization $h_i =: h_{i,1}h_{i,2}$ of some generator, we split I_l efficiently into two split ideals $I_1 = \langle h_0, \ldots, h_{i-1}, h_{i,1}, \ldots, h_l\rangle$ and $I_2 = \langle h_0, \ldots, h_{i-1}, h_{i,2}, \ldots, h_l\rangle$ (a key property of split ideals).

Our simplified algorithm gives rise to a different roots-tree than [CGRW18], and helps us to observe some key properties of our roots-tree (Section 3.3). Using these properties, we could generalize the multiplicity argument of [BLQ13] (also improvement of [KRRZ19, Lemma 3.6]) which in turn helped in bounding the number of clusters and tree size.

We feel that our idea is quite natural as Algorithm 1 easily extends to unramified extensions, as was the case with [BLQ13], and this in turn gives the count of all basic-irreducible factors as well. Moreover, our methods apply to the p-adic context (as they work for arbitrary k).

2 Preliminaries

Here we introduce our main tool - 'split ideals'. Proofs for this section have been moved to Section B. Basic introduction to Galois rings (i.e. non-prime characteristic analog of finite fields), Hensel lifting, randomized factoring over finite fields, etc. have been moved to Section A.

We will be given a univariate polynomial $f(x) \in \mathbb{Z}[x]$ of degree d and a prime power p^k (for a prime p and a positive integer $k \in \mathbb{N}$). Wlog, we assume that f is monic over $\mathbb{F}_p$.

A tuple of variables $(x_0, \ldots, x_l)$ will be denoted by $\bar{x}_l$. Often, an $(l+1)$-variate polynomial $a(x_0, x_1, \ldots, x_l)$ will be written as $a(\bar{x}_l)$, and the polynomial ring $\mathbb{F}_p[x_0, \ldots, x_l]$ as $\mathbb{F}_p[\bar{x}_l]$.

We denote the ring $\mathbb{Z}/\langle p^k \rangle$ by R (ring $R/\langle p \rangle$ is the same as field $\mathbb{F}_p$). An element $a \in R$ can be seen in its p-adic representation as $a = a_0 + pa_1 + \ldots + p^{k-1}a_{k-1}$, where $a_i \in \mathbb{F}_p$ for $i \in \{0, \ldots, k-1\}$.

$\mathcal{Z}_R(g) := \{r \in R \mid g(r) \equiv 0 \bmod p^k\}$ denotes the *zeroset* of a polynomial $g(x) \in R[x]$.

Zeroset of an ideal $I \subseteq \mathbb{F}_p[x_0, \ldots, x_l]$ is defined as the intersection of zeroset of all polynomials in I, $\mathcal{Z}_{\mathbb{F}_p}(I) := \{\bar{a} = (a_0, \ldots, a_l) \in (\mathbb{F}_p)^{l+1} \mid g(\bar{a}) \equiv 0 \bmod p, \forall g \in I\}$.

We will heavily use ideals of the form $I := \langle h_0(\bar{x}_0), h_1(\bar{x}_1), \ldots, h_l(\bar{x}_l) \rangle$ satisfying the condition—for any $i \in [l+1]$ and $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(\langle h_0(\bar{x}_0), h_1(\bar{x}_1), \ldots, h_{i-1}(\bar{x}_{i-1}) \rangle)$, polynomial $h_i(\bar{a}, x_i)$ splits completely into distinct linear factors. They are formally defined as:

Definition 4 (Split ideal)**.** *We will call a polynomial* monic wrt x *if the leading-coefficient is one. Given $f(x) \in R[x]$, an ideal I, in $\mathbb{F}_p[\bar{x}_l]$, is called a* split ideal wrt $f \bmod p^k$ *if,*
1) I is a triangular ideal of length $l+1$*, meaning: $I =: \langle h_0(\bar{x}_0), h_1(\bar{x}_1), \ldots, h_l(\bar{x}_l) \rangle$, for some $0 \le l \le k-1$; $h_i(\bar{x}_i) \in \mathbb{F}_p[\bar{x}_i]$ is monic wrt x_i, for all $i \in \{0, \ldots, l\}$,*
2) $|\mathcal{Z}_{\mathbb{F}_p}(I)| = \prod_{i=0}^{l} \deg_{x_i}(h_i)$, and
3) $\forall (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I)$, $f(a_0 + pa_1 + \ldots + p^l a_l) \equiv 0 \bmod p^{l+1}$.

The length of I *is $l+1$ and its* degree *is* $\deg(I) := \prod_{i=0}^{l} \deg_{x_i}(h_i)$.

Split ideal I relates to possible roots of $f \bmod p^k$. Since f, p, k are fixed, we will call I a *split ideal*. The definition of a split ideal implies that its roots represent a set of "potential" roots of f, i.e. roots of f modulo some p^{l+1} for $0 \le l < k$. Restriction of a split ideal is also a split ideal.

Lemma 5 (Restriction of a split ideal)**.** *Let $I_l := \langle h_0(\bar{x}_0), \ldots, h_l(\bar{x}_l) \rangle$ be a split ideal in $\mathbb{F}_p[x_0, \ldots, x_l]$, then ideal $I_j := \langle h_0(\bar{x}_0), \ldots, h_j(\bar{x}_j) \rangle$ is also a split ideal in $\mathbb{F}_p[x_0, \ldots, x_j]$, for all $0 \le j \le l$.*

Further, we show that a split ideal I can be decomposed in terms of its zeros.

Lemma 6 (Split ideal structure)**.** *A split ideal $I \subseteq \mathbb{F}_p[x_0, \ldots, x_l]$ can be decomposed as $I = \bigcap_{\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I)} I_{\bar{a}}$, where each $I_{\bar{a}} := \langle x_0 - a_0, \ldots, x_l - a_l \rangle$ corresponds to root $\bar{a} =: (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I)$. By Chinese remainder theorem, $R/I = \bigoplus_{\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I)} R/I_{\bar{a}}$.*

Let $I =: \langle h_0(\bar{x}_0), h_1(\bar{x}_1), \ldots, h_l(\bar{x}_l) \rangle$ be a split ideal. Suppose some h_i factors as $h_i(\bar{x}_i) = h_{i,1}(\bar{x}_i) \ldots h_{i,m}(\bar{x}_i)$. Define $I_j := \langle h_0(\bar{x}_0), \ldots, h_{i-1}(\bar{x}_{i-1}), h_{i,j}(\bar{x}_i), h_{i+1}(\bar{x}_{i+1}), \ldots, h_l(\bar{x}_l) \rangle$, for $j \in [m]$. The following corollary of Lemma 6 is evident because root-sets of I_j partition the root-set of I.

Corollary 7 (Splitting split ideals)**.** *Let $I = \langle h_0(\bar{x}_0), \ldots, h_l(\bar{x}_l) \rangle$ be a split ideal of $\mathbb{F}_p[x_0, \ldots, x_l]$. Let some $h_i(\bar{x}_i)$ factor as $h_i(\bar{x}_i) = h_{i,1}(\bar{x}_i) \ldots h_{i,m}(\bar{x}_i)$.*

Then, $I = \bigcap_{j=1}^{m} I_j$, where each $I_j := \langle h_0(\bar{x}_0), \ldots, h_{i-1}(\bar{x}_{i-1}), h_{i,j}(\bar{x}_i), h_{i+1}(\bar{x}_{i+1}), \ldots, h_l(\bar{x}_l) \rangle$ is a split ideal.

We call a split ideal $I_l := \langle h_0, \ldots, h_l \rangle$ to be *maximal split ideal* if,
1) for any $\bar{a} = (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I_l)$, $g(x) := f(a_0 + pa_1 + \ldots + p^l a_l + p^{l+1}x)$ vanishes identically mod p^k,
2) the restriction $I_{l-1} := \langle h_0, \ldots, h_{l-1} \rangle$ does not follow the previous condition.

Lemma 8 (Roots represented by a root of maximal split ideal)**.** *Let I be a maximal split ideal of length $l+1$, then a zero $\bar{a} = (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I)$ maps to exactly p^{k-l-1} zeros of f in $\mathcal{Z}_R(f)$. We will say that these p^{k-l-1} roots of f are* represented *by $\bar{a}$.*

3 Proof of Theorem 1

The algorithm to compute a compact data-structure which stores roots of $f \bmod p^k$ will be described in Section 3.1. Algorithm's correctness will be proved in Section 3.2, which involves studying the algebraic structure underlying the algorithm. Its efficiency will be shown in Section 3.3, by devising an auxiliary structure called roots-tree and the important notion of 'degree of a node'.

3.1 Algorithm to implicitly partition the root-set of $f(x) \bmod p^k$

We describe our algorithm in this section. It takes a monic univariate polynomial $f(x) \in \mathbb{Z}[x]$ of degree d and a prime-power p^k as input (in binary), and outputs a list of at most d maximal split ideals whose roots partition the root-set of f modulo p^k.

A maximal split ideal $I_j =: \langle h_0(\bar{x}_0), \ldots, h_l(\bar{x}_l) \rangle$ has $|\mathcal{Z}_{\mathbb{F}_p}(I_j)| = \prod_{i=0}^{l} \deg_{x_i}(h_i)$ zeros, and each such zero 'represents' p^{k-l-1} actual zeros of $f \bmod p^k$ (Lemma 8). Thus, this algorithm gives an exact count on the number of zeros of f in R.

Overview of Algorithm 1: Since any root of $f \bmod p^k$ is an extension of a root modulo p, the algorithm starts by initializing a stack S with the ideal $I := \langle h_0(x_0) \rangle$, where $h_0(x_0) := \gcd(x_0^p - x_0, f(x_0))$. This is a split ideal containing all the roots of $f \bmod p$. By a *lift* $\hat{I} \subset R[x_0]$ of I, we mean the ideal generated by the generator $\{h_0\}$ when viewed as a polynomial in $R[x_0]$ (i.e. char p^k).

At every intermediate iteration (Steps $4 - 21$), we *pop* a split ideal from the stack and *try* to increase the precision of its root-set (equivalently, lengthen the split ideal). This step mostly results in two cases: either we succeed and get a split ideal whose root-set has increased precision (Step 18) by a new placeholder x_{l+1}, or the split ideal factors into more split ideals increasing the size of the stack S (Steps $10, 14, 20$). We update the relevant 'part of f' to $f_I(\bar{x}_l, x_{l+1} + px) \bmod \hat{J}$ (J is the new split ideal) that we carry around with each split ideal. This helps in efficiently increasing the precision of roots in the next iteration. Otherwise, computing $f\left(x_0 + px_1 + \cdots + p^l x_l + p^{l+1}x\right)/p^\alpha \bmod I$ is too expensive, in Step 6, due to the underlying degree-d $(l+1)$-variate monomials blowup.

If we reach a maximal split ideal (Step 7), it is moved to a list $\mathcal{L}$. Sometimes the split ideal cannot be extended and we get a *dead-end* (Step 16). The size of the stack decreases when we get a maximal split ideal or a dead-end. The algorithm terminates when stack becomes empty. List $\mathcal{L}$ contains maximal split ideals which partition, and cover, the root-set of f (implicitly). This becomes our output.

The main intuition behind our algorithm: If two roots of a split ideal (representing potential roots of f) give rise to different number of roots of f, the split ideal will get factored further. Though not at all apparent immediately, we will show that the algorithm takes only polynomial number of steps (Section 3.3).

We will use four subroutines to perform standard ring arithmetic modulo split ideals; they are described in the Appendices C & D.

1. Modify f (Steps 3, 18, 20) whenever pushing in the stack (Lemma 30 & 31).
2. REDUCE$(a(\bar{x}_l), J_l)$ gives the reduced form of a mod triangular ideal J_l (over a Galois ring).
3. TEST-ZERO-DIV$(a(\bar{x}_l), I_l)$ either reports that a is a not a zero-divisor modulo triangular ideal I_l or outputs a non-trivial factorization of one of the generators of I_l when true.
4. GCD$(a(\bar{x}_l, x), b(\bar{x}_l, x), I_l)$ either successfully computes a monic gcd, wrt x, of two multivariates modulo a triangular ideal I_l, or encounters a zerodivisor in intermediate computation (outputting $False$ and a non-trivial factorization of one of the generators of I_l).

Algorithm 1 Root-counting mod p^k

1: Let $\mathcal{L} = \{\}$ be a list and $S = \{\}$ be a stack (both initially empty).
2: Let $\tilde{f}(x_0) := f(x_0) \bmod p$ for a monic univariate $\tilde{f} \in \mathbb{F}_p[x_0]$ of degree d.
3: [**Initializing the stack S**] Let $h_0(x_0) := \gcd(\tilde{f}(x_0), x_0^p - x_0)$, $I := \langle h_0 \rangle$, $\hat{I} \subseteq R[x_0]$ be a lift of I. Compute $f_I(x_0, x) := f(x_0 + px) \bmod \hat{I}$ using Lemma 30. Update $S \leftarrow push((\{h_0\}, f_I))$.
4: **while** S is not empty **do**
5: $S_{top} \leftarrow pop(S)$. Let $S_{top} = (\{h_0(x_0), \ldots, h_l(x_0, \ldots, x_l)\}, f_I(\bar{x}_l, x))$ where $I = \langle h_0, \ldots, h_l \rangle \subseteq \mathbb{F}_p[\bar{x}_l]$ is a split ideal. Let $\hat{I} \subseteq R[x_0, \ldots, x_l]$ be a lift of I.
6: [**Valuation computation**] Compute $\alpha \in \mathbb{N}$ and $g \in R[\bar{x}_l, x]$ such that $f_I \equiv p^\alpha g(\bar{x}_l, x) \bmod \hat{I}$ and $p \not| g \bmod \hat{I}$.
7: [**Maximal split ideal found**] **if**$(\alpha \geq k)$ **then** update List $\mathcal{L} \leftarrow \mathcal{L} \cup \{I\}$. Go to Step 4.
8: Let $\tilde{g} := g(\bar{x}_l, x) \bmod I$ be the polynomial in $\mathbb{F}_p[\bar{x}_l, x]$, and let $g_1(\bar{x}_l)$ be the leading coefficient of $\tilde{g}(\bar{x}_l, x)$ wrt x.
9: **if** TEST-ZERO-DIV$(g_1(\bar{x}_l), I) = True$ **then**
10: TEST-ZERO-DIV$(g_1(\bar{x}_l), I)$ returns a factorization $h_i(\bar{x}_i) =: h_{i,1}(\bar{x}_i)h_{i,2}(\bar{x}_i)\ldots h_{i,m}(\bar{x}_i) \bmod I_{i-1}$ of some generator $h_i(\bar{x}_i)$ of I. Go to Step 20.
11: **end if**
[**Filter out distinct virtual $\mathbb{F}_p$-roots by taking gcd with $x^p - x$**]
12: Recompute $\tilde{g} := g(\bar{x}_l, x) \cdot g_1(\bar{x}_l)^{-1} \bmod I$ (Lemmas 29, 28). Compute x^p by repeatedly squaring and reducing modulo the triangular ideal $I + \langle \tilde{g} \rangle$ (Algorithm 2 and Lemma 28). This yields $\tilde{h}_{l+1}(\bar{x}_l, x) := x^p - x \bmod I$ in a reduced form.
13: **if** GCD$(\tilde{g}, \tilde{h}_{l+1}, I) = False$ **then**
14: The call GCD$(\tilde{g}, \tilde{h}_{l+1}, I)$ returns factorization $h_i(\bar{x}_i) = h_{i,1}(\bar{x}_i)h_{i,2}(\bar{x}_i)\ldots h_{i,m}(\bar{x}_i) \bmod I_{i-1}$ of a generator $h_i(\bar{x}_i)$ of I. Go to Step 20.
15: **else if** $\tilde{g}$ and $\tilde{h}_{l+1}$ are coprime **then**
16: [**Dead End**] The ideal I cannot grow more, go to Step 4.
17: **else**
18: [**Grow the split ideal** I] Here $\gcd_x(\tilde{g}, \tilde{h}_{l+1}) \bmod I$ is non-trivial, say $h_{l+1}(\bar{x}_l, x)$ (monic wrt x). Substitute x by x_{l+1} in $h_{l+1}(\bar{x}_l, x)$ and update $J \leftarrow I + \langle h_{l+1}(\bar{x}_{l+1}) \rangle$.
Let $\hat{J} \subseteq R[x_0, \ldots, x_{l+1}]$ be a lift of J. Substitute x by $x_{l+1} + px$ in $f_I(\bar{x}_l, x)$, and compute $f_J(\bar{x}_{l+1}, x) := f_I(\bar{x}_l, x_{l+1} + px) \bmod \hat{J}$ using Lemma 30. Update $S \leftarrow push((\{h_0, \ldots, h_{l+1}\}, f_J))$, and go to Step 4.

19: **end if**

20: [**Factoring split ideals**] We have a factorization $h_i(\bar{x}_i) = h_{i,1}(\bar{x}_i)h_{i,2}(\bar{x}_i)\dots h_{i,m}(\bar{x}_i) \bmod I_{i-1}$ of a generator h_i of I. Push S_{top} back in stack S. For every entry $(U, f_{\langle U\rangle}) \in S$, where $h_i(\bar{x}_i)$ appears in U, find m (smaller) split ideals U_j (using Corollary 7); using Lemma 31 compute $f_{\langle U_j\rangle}$ and push $(U_j, f_{\langle U_j\rangle})$ in S, for $j \in [m]$.

21: **end while**

22: Return $\mathcal{L}$ (the list of maximal split ideals partitioning the root-set $\mathcal{Z}_R(f)$).

3.2 Correctness of Algorithm 1

Our main goal is to prove the following result about partitioning of root-set.

Theorem 9 (Algo 1 partitions $\mathcal{Z}_R(f)$)**.** *Algorithm 1 yields the structure of the root-set $\mathcal{Z}_R(f)$ through a list data structure $\mathcal{L}$ (a collection of maximal split ideals $I_1, \dots, I_n$) which partitions the zeroset $\mathcal{Z}_R(f) =: \bigsqcup_{j\in[n]} S_j$, where S_j is the set of roots of $f \bmod p^k$ represented by $\mathcal{Z}_{\mathbb{F}_p}(I_j)$.*

Later, we will show a surprising property: $n \leq d$ (Section 3.3).

Proof of Theorem 9. Lemma 12 states that at any point, Stack S only contains split ideals which have disjoint root sets. Lemma 13 assures that Algorithm 1 terminates on every input. So from Lemmas 12, 13 and the definition of maximal split ideal, it is clear that Algorithm 1 returns a list $\mathcal{L}$ containing maximal split ideals $I_1, \dots, I_n$, for $n \in \mathbb{N}$. Further, we show:
1) The root-set of I_j $(1 \leq j \leq n)$ yields a subset S_j of $\mathcal{Z}_R(f)$, and they are pairwise disjoint.
2) Given a root $r \in \mathcal{Z}_R(f)$, there exists j such that r is represented by a root in $\mathcal{Z}_{\mathbb{F}_p}(I_j)$.

For the first part, root-sets for different maximal split ideals I_j are pairwise disjoint because of Lemma 12. Each of these root-set yields a subset of the zeroset of $f \bmod p^k$ (follows from the definition of maximal split ideal).

For the second part, let $r =: \sum_{i=0}^{k-1} r_i p^i$ be a root in $\mathcal{Z}_R(f)$. Stack S was initialized by the split ideal $\langle h_0 := \gcd(f(x_0) \bmod p, x_0^p - x_0)\rangle$; so $r_0 \in \mathcal{Z}_{\mathbb{F}_p}(I_0)$, as $f(r_0) \equiv f(r) \equiv 0 \bmod p$.

Assume that I_0 is not a maximal split ideal (otherwise we are done). Applying Lemma 14, there must exist an I_1 whose root-set contains (r_0, r_1). Repeated applications of Lemma 14 show that we will keep getting split ideals of larger lengths, partially representing r; finally, reaching a maximal split ideal (say I_j) fully representing r.

We showed that each root r of $f \bmod p^k$ is represented by a *unique* maximal split ideal I, given by Algorithm 1, and they collectively represent exactly the roots of f modulo p^k. Hence, root-sets of ideals in $\mathcal{L}$ partition the zeroset $\mathcal{Z}_R(f)$. □

Now, let us see the properties of our algorithm which go in proving Theorem 9. Given a polynomial $g(\bar{x}_l) \in \mathbb{F}_p[\bar{x}_l]$ and an element $\bar{a} \in \mathbb{F}_p^l$, consider the *projection* $g_{\bar{a}}(x_l) := g(\bar{a}, x_l)$. Using Chinese remainder theorem (Lemma 6) we easily get the following degree condition. (Here, lc_x refers to the leading coefficient wrt variable x.)

Claim 10. *Let I be a split ideal of $\mathbb{F}_p[\bar{x}_{l-1}]$ and $g \in \mathbb{F}_p[\bar{x}_l]$. Then, $lc_{x_l}(g)$ is unit mod I iff $\forall \bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I)$, $\deg(g_{\bar{a}}(x_l)) = \deg_{x_l}(g(\bar{x}_l) \bmod I)$.*

Chinese remaindering also gives us a gcd property under projections.

Lemma 11. *Let $w(\bar{x}_l), z(\bar{x}_l) \in \mathbb{F}_p[\bar{x}_l]$ and $I_{l-1} \subseteq \mathbb{F}_p[\bar{x}_{l-1}]$ be a split ideal. Suppose Algorithm 4 succeeds in computing gcd of w and z mod I_{l-1}: define $h(\bar{x}_l) := \mathrm{GCD}(w(\bar{x}_l), z(\bar{x}_l), I_{l-1})$. Then, for all $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I_{l-1})$: $h_{\bar{a}}(x_l)$ equals $\gcd(w_{\bar{a}}(x_l), z_{\bar{a}}(x_l))$ up to a unit multiple (in $\mathbb{F}_p^*$).*

Proof. Lemma 33 proves, $h(\bar{x}_l)$ is a monic polynomial mod I_{l-1}, s.t., $h|w$ and $h|z$ (mod I_{l-1}). Fix $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I_{l-1})$. Since $h_{\bar{a}}(x_l) \not\equiv 0 \bmod p$ ($\because h$ is monic), restricting $\bar{x}_{l-1}$ to $\bar{a}$ gives $h_{\bar{a}}|w_{\bar{a}}$ and $h_{\bar{a}}|z_{\bar{a}}$, showing $h_{\bar{a}}| \gcd(w_{\bar{a}}, z_{\bar{a}})$, in $\mathbb{F}_p[x_l]$.

Lemma 33 also shows that there exists $u, v \in (\mathbb{F}_p[\bar{x}_{l-1}]/I_{l-1})[x_l]$, such that, $h = uw + vz$. Restricting first l co-ordinates to $\bar{a}$, we get $h_{\bar{a}} = u_{\bar{a}} w_{\bar{a}} + v_{\bar{a}} z_{\bar{a}}$. This equation implies $\gcd(w_{\bar{a}}, z_{\bar{a}})|h_{\bar{a}}$. Thus, we get an equality up to a unit multiple. □

Let $I \subseteq \mathbb{F}_p[\bar{x}_i], J \subseteq \mathbb{F}_p[\bar{x}_j]$ be two split ideals (say $i \leq j$). I and J are called *prefix-free* iff $\nexists\ \bar{a} = (a_0, a_1, \ldots, a_i) \in \mathcal{Z}_{\mathbb{F}_p}(I),\ \bar{b} = (b_0, b_1, \ldots, b_j) \in \mathcal{Z}_{\mathbb{F}_p}(J) : a_k = b_k\ \forall k \leq i$.

(Note that it may still happen that $(a_0, \ldots, a_{i-1}) = (b_0, \ldots, b_{i-1})$ above.)

Our next lemma shows an invariant about Algorithm 1.

Lemma 12 (Stack contents)**.** *Stack S in Algorithm 1 satisfies following conditions at every point:*
1) $l < k$ and in Step 6, $\alpha > l$.
2) All ideals in S are split ideals.
3) Any two ideals in S are prefix-free.

Proof. We first prove the invariant 1. Step 6 defines g via f_I as, $f_I =: p^\alpha g(\bar{x}_l, x) \bmod \hat{I}$. Looking at the f_I analogues pushed in Steps 3, 18, 20, one easily deduces the invariants:
$f\left(\sum_{0\leq i\leq l} x_i p^i + x p^{l+1}\right) \equiv f_I(\bar{x}_l, x) \bmod \hat{I}$, and
$f\left(\sum_{0\leq i\leq l} x_i p^i\right) \equiv 0 \bmod \hat{I} + \langle p^{l+1}\rangle$.

Thus, $f\left(\sum_{0\leq i\leq l} x_i p^i\right) \equiv p^\alpha g(\bar{x}_l, x) \equiv 0 \bmod \hat{I} + \langle p^{l+1}\rangle$. Since, $p \nmid g \bmod \hat{I}$, we deduce $\alpha > l$. Moreover, by Step 7 we know that $l < k$ throughout the algorithm.

There are three ways in which a new ideal is added to stack S. We show below that the invariant is maintained in all three cases.

(Step 3) S is initialized with the ideal $I = \langle h_0(x_0)\rangle \subseteq \mathbb{F}_p[x_0]$. The triangular ideal I is a split ideal, because $|\mathcal{Z}_{\mathbb{F}_p}(I)| = \deg_{x_0}(h_0)$ and its root are all the distinct roots of $f(x_0) \bmod p$.

(Step 20) Ideal I_l is popped from S, and some generator h_i of I_l splits. In this case, we update S with the corresponding factors of any $(U, f_{\langle U\rangle}) \in S$, wherever currently U has h_i. Corollary 7 shows that the factors of U are split ideals themselves, and their root-sets partition that of U. Thus, these root-sets are prefix-free among themselves. Moreover, they are prefix-free with any other ideal J appearing in S, because U was prefix-free with J.

(Step 18) Ideal I_l is popped, it grows to I_{l+1} by including $h_{l+1}(\bar{x}_l, x) = \gcd_x(\tilde{g}(\bar{x}_l, x), x^p - x) \bmod I_l$ ($\tilde{g}$ is defined in Step 8). First (resp. third) condition for I_{l+1} being a split ideal follows from the definition of $\tilde{g}$ (resp. h_{l+1}).

For the second condition for I_{l+1} being a split ideal, fix a particular root $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I_l)$. Using Lemma 11, the projection $h_{l+1,\bar{a}}(x)$ equals $\gcd(\tilde{g}_{\bar{a}}(x), x^p - x)$ (up to a unit multiple). By Lemma 33, h_{l+1} is monic mod I_l; giving $\deg(h_{l+1,\bar{a}}) = \deg_{x_{l+1}}(h_{l+1})$. Since $h_{l+1}|x^p - x$, there are exactly $\deg_x(h_{l+1})$-many $a_{l+1} \in \mathbb{F}_p$, such that $h_{l+1,\bar{a}}(a_{l+1}) \equiv 0 \bmod p$. So, every root $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I_l)$ can be extended to $\deg_x(h_{l+1})$-many roots; giving $|\mathcal{Z}_{\mathbb{F}_p}(I_{l+1})| = \deg_x(h_{l+1}) \cdot \prod_{i=0}^{l} \deg_{x_i}(h_i)$. This makes I_{l+1} a split ideal.

I_{l+1} remains prefix-free with any other ideal J of S, because roots of I_{l+1} are extension of roots of I_l (recall: I_l was prefix-free with J and it was popped out of S).

This proves all the invariants for the stack S. □

Using the invariant, we prove that Algorithm 1 terminates on any input.

Lemma 13. *Algorithm 1 finishes in finite number of steps for any $f \in \mathbb{Z}[x]$ and a prime power p^k.*

Proof. We show that the number of iterations in Algorithm 1 are finite. Assume that all the ideals which result in a dead-end are moved to a list D; say C is the disjoint union of all ideals in S, $\mathcal{L}$ and D. Whenever a split ideal I from S is moved to $\mathcal{L}$ or D, the underlying roots (of I) stop extending to the next precision. Togetherwith Lemma 12, we deduce that in fact all the ideals in C are prefix-free. Now by Step 18, and the rate of growth of split ideals up to length $l+1 \leq k$, we get a lazy estimate of $|C| \leq \min(d^k, p^k)$.

Let $\mathrm{len}(I)$ denote the length of an ideal I, it is bounded by k. Notice that factoring/growing an ideal increases $\sum_{I\in C} \mathrm{len}(I)$; and getting a maximal split ideal/ dead-end increases $|\mathcal{L}| + |D|$. Thus, every iteration of the algorithm strictly increases the quantity $(\sum_{I\in C} \mathrm{len}(I)) + |\mathcal{L}| + |D|$. By the estimate on $|C|$, all the terms in this quantity are bounded; thus, the number of iterations are finite. □

The following lemma shows: if we see a restriction of $r \in \mathcal{Z}_R(f)$ (say, up to length $l+1$) at some point in Algorithm 1, we will again see its restriction of length $l+2$ at a later point in the algorithm.

Lemma 14 (Getting roots with more precision)**.** *Assume that at some time (say t), Algorithm 1 pops an ideal I of length $l+1$, that is not yet a maximal split ideal. Let $\bar{a} = (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I)$ partially represent a "root" $r =: \sum_{0\leq i\leq l+1} a_i p^i$ such that $f(r) \equiv 0 \mod p^{l'}$, but $f\left(\sum_{0\leq i\leq l} a_i p^i + x p^{l+1}\right) \not\equiv 0 \mod p^{l'}$, for some $l+2 \leq l' \leq k$. Then, there exists a time $t' > t$, when stack S will pop an ideal J of length $l+2$, such that, $(\bar{a}, a_{l+1}) \in \mathcal{Z}_{\mathbb{F}_p}(J)$.*

Proof. We again consider three possible situations.

(Step 18) Ideal I grows to another split ideal, say J. Notice, J is obtained by adding $h_{l+1} := \mathrm{GCD}(g(\bar{x}_l, x), x^p - x) \bmod I$ to I (setting $x \mapsto x_{l+1}$).

Step 6 defines g via f_I as, $f_I =: p^\alpha g(\bar{x}_l, x) \bmod \hat{I}$. Looking at the f_I analogues pushed in Steps 3, 18, 20, one can deduce the invariant: $f\left(\sum_{0\leq i\leq l} x_i p^i + x p^{l+1}\right) \equiv f_I(\bar{x}_l, x) \bmod \hat{I}$.

Now, let us project to (suitable integral lifts of) $\bar{a}$ and consider $f\left(\sum_{0\leq i\leq l} a_i p^i + x p^{l+1}\right) \equiv f_I(\bar{a}, x) \equiv p^\alpha g(\bar{a}, x) \bmod \hat{I}$. By Step 9, and Claim 10, we are assured that $g(\bar{a}, x)$, $g(\bar{x}_l, x) \bmod I$ are equi-degree (wrt x). Thus, by non-maximality hypothesis we have $\alpha < l'$. Hypothesis tells us that $f\left(\sum_{0\leq i\leq l+1} a_i p^i\right) \equiv 0 \mod p^{l'}$. So, by the previous paragraph, $p^\alpha g(\bar{a}, a_{l+1}) \equiv 0 \mod p^{l'}$. Whence, $g(\bar{a}, a_{l+1}) \equiv 0 \bmod p$. Clearly, $a_{l+1}^p - a_{l+1} \equiv 0 \bmod p$. Thus, $h_{l+1}(\bar{a}, a_{l+1}) \equiv 0 \bmod p$. So $(\bar{a}, a_{l+1})$ is a root of J.

(Step 16) Proof of the previous case shows that $h_{l+1}(\bar{a}, x)$ has degree at least 1, so I could not result in a *dead-end*.

(Step 20) Ideal I factors into (smaller) split ideals. In this case, $\bar{a}$ will be included in exactly one of those ideals (by Corollary 7). This ideal will be handled later in the algorithm and will give an ideal J with $(\bar{a}, a_{l+1})$ as root. □

3.3 Time complexity of Algorithm 1— introducing roots-tree RT

We know that Algorithm 1 takes finite amount of time and terminates (Lemma 13). To show that it is efficient, note that the time complexity of the algorithm can be divided into two parts.

1) Number of iterations taken by Algorithm 1, which is clearly bounded by the number of updates on Stack S in the algorithm.

2) Time taken by the various algebraic operations in one iteration of the algorithm: reduction by a triangular ideal, valuation computation modulo a split ideal, testing if some polynomial is a zerodivisor modulo a split ideal, performing repeated squaring modulo a triangular ideal and computing gcd of two multi-variates modulo a split ideal.

For the purpose of bounding iterations, we define a 'virtual' tree, called *roots-tree (RT)*, which essentially keeps track of the updates on Stack S. We will map a node $N = (I, f_I)$ in roots-tree to the element (I, f_I) in stack S. Each *push* will create a new node in RT. The nodes are *never* deleted from RT.

Construction of roots-tree (RT): Denote the root of RT by $N_{\langle 0\rangle} := (\langle 0\rangle, f_{\langle 0\rangle} := f(x))$. Add a child node N_{I_0} to the root corresponding to the initialization of Stack S by (I_0, f_{I_0}), where $I_0 := \langle h_0(x_0)\rangle$ (label the edge h_0 in RT).

If, at some time t, the algorithm pops $(I_{l-1}, f_{I_{l-1}})$ from S then the *current node* in RT will be the leaf node $N_{I_{l-1}} = (I_{l-1}, f_{I_{l-1}})$. We map the updates on stack S to RT as follows:

(Step 18) If ideal I_{l-1} *grows* to $I_l := I_{l-1} + \langle h_l\rangle$ and (I_l, f_{I_l}) is pushed in S, then create a child of $N_{I_{l-1}}$ in RT using an edge labelled h_l (label the node $N_{I_l} := (I_l, f_{I_l})$).

(Steps 7, 16) If the algorithm reached *dead-end* (no update in stack S or list $\mathcal{L}$), then add a child labelled $\mathcal{D}$ to node $N_{I_{l-1}}$. It indicates a dead-end at the current branch. Analogously, if the algorithm finds a *maximal split ideal*, we add a child labelled $\mathcal{M}$ to Node $N_{I_{l-1}}$ (indicating I_{l-1} is a maximal split ideal).

(Step 20) Suppose, processing of length-l split ideal I_{l-1} results in factoring each ideal U in S, containing h_i, to m split ideals. We describe the *duplication process* for a particular U (repeat it for each split ideal containing h_i).

Let U_{i-1} be the length-i restriction of U. First, we move to the ancestor node $N_{U_{i-1}} := (U_{i-1}, f_{U_{i-1}})$ of N_U. Make m copies of the sub-tree at Node $N_{U_{i-1}}$, each of them attached to $N_{U_{i-1}}$ by edges labelled with $h_{i,1}, \ldots, h_{i,m}$ respectively. The copy of each old node $N = (V, f_V)$, in sub-tree corresponding to $h_{i,j}$, will be relabelled with (V_j, f_{V_j}) corresponding to the factor split ideal V_j of V and the newly computed f_{V_j}.

This step does not increase the height of the tree, though it increases the size.

For the rest of this section, RT denotes the final roots-tree created at the end of the above process.

Properties of RT: We state some easy properties of RT, which will help us in analyzing the time complexity.

1) By construction, size of the roots-tree increases at every iteration. We never delete a node or an edge (though relabelling might be done). So, the size of RT bounds the number of iterations taken by Algorithm 1.

2) Consider a node $N_I =: (I, f_I)$ in RT. Here $f_I(\bar{x}_l, x) \in R[\bar{x}_l, x]$, and let $g_I \in R[\bar{x}_l, x]$ be defined as in Algorithm 1, $g_I := f_I(\bar{x}_l, x)/p^\alpha \bmod \hat{I}$, where $p^\alpha \mid\mid f_I \mod \hat{I}$, and $\hat{I}$ is a lift of I over R. Then, $g_I \bmod I$ is a nonzero polynomial over $\mathbb{F}_p$.

3) For each node $N_I =: (I, f_I(\bar{x}_l, x))$ and its child $N_J =: (J, f_J(\bar{x}_{l+1}, x))$, we have the relation, $f_J = f_I(\bar{x}_l, x_{l+1} + px) \bmod \hat{J}$.

Bounding $|RT|$: To bound the size of RT, we define a parameter for a node N of RT, called the *degree* of the node N and denoted by $[N]$.

Definition 15 (Degree of a node in RT)**.** *Degree of leaves $\mathcal{D}$ resp. $\mathcal{M}$ is defined to be* 1.

Let $N_I =: (I, f_I)$ be a node corresponding to a split ideal $I \subseteq \mathbb{F}_p[\bar{x}_l]$, where $f_I(\bar{x}_l, x)$ belongs to $R[\bar{x}_l, x]$. Let $p^\alpha \mid\mid f_I \bmod \hat{I}$ and $g_I(\bar{x}_l, x) := f_I/p^\alpha \bmod \hat{I}$. Except, $g_I := 0$ if $\alpha \geq k$.

Then, the degree of N is defined as, $[N] := \max(1, \deg_x(g_I \bmod I) \times \deg(I))$.

We remark that for the root node, $N_{\langle 0\rangle} = (\langle 0\rangle, f_{\langle 0\rangle} := f(x))$ we will set $\deg(\langle 0\rangle) := 1$. Then, it is clear by the general definition of degree that $[N_{\langle 0\rangle}] = d$ $(= \deg(f) \geq 1)$.

We show that the degree of a parent node bounds the sum of the degree of its children.

Lemma 16 (Degree distributes in RT)**.** *Let N be a node in roots-tree RT and $des(N)$ denote the set of all children of N. Then, $[N] \geq \sum_{C \in des(N)} [C]$.*

So, the sum of the degrees of all nodes, at any level l, is at least the sum of the degrees of all nodes at level $l+1$.

Proof. Let $N = (I, f_I)$, where $I = \langle h_0, \ldots, h_l\rangle$ and $f_I(\bar{x}_l, x) \in R[\bar{x}_l, x]$. Define $\tilde{g}_I \in \mathbb{F}_p[\bar{x}_l, x]$ as $\tilde{g}_I := g_I(\bar{x}_l, x) \bmod I$. Assume $\alpha < k$, otherwise we are done. So, $g_I \bmod I$ is nontrivial wrt x; by Step 9 (failure) and Claim 10, we get,

$$\forall \bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I) : \ \deg_x(\tilde{g}_I \bmod I) \ = \ \deg_x(\tilde{g}_I(\bar{a}, x)) \ . \tag{1}$$

Recall $h_{l+1}(\bar{x}_l, x) := \gcd(\tilde{g}_I(\bar{x}_l, x), x^p - x)$.Let C be a child node of N in RT such that $C =: (J_C, f_{J_C})$, where $J_C =: I + \langle h_{l,C}(\bar{x}_{l+1})\rangle$ and $f_{J_C}(\bar{x}_{l+1}, x) := f_I(\bar{x}_l, x_{l+1} + px) \bmod \hat{J}_C$. This gives us the factorization $h_{l+1}(\bar{x}_l, x) = \prod_{C \in \mathrm{des}(N)} h_{l,C}(\bar{x}_l, x) \bmod I$ (Step 20, and 'duplication step' when we constructed RT). Again,

$$\forall \bar{b} \in \mathcal{Z}_{\mathbb{F}_p}(J_C) : \ \deg_x(\tilde{g}_{J_C} \bmod J_C) \ = \ \deg_x(\tilde{g}_{J_C}(\bar{b}, x)) \ . \tag{2}$$

If $g_{J_C} =: f_{J_C}/p^{v'} \bmod \hat{J}_C$ for some $v' \in \mathbb{N}$, by property 3 of RT, we have $g_{J_C} = f_I(\bar{x}_l, x_{l+1} + px)/p^{v'} \bmod \hat{J}_C$.

By definition, $[N] = \deg(I) \cdot \deg_x(\tilde{g}_I)$ and $[C] = \deg(J_C) \cdot \deg_x(\tilde{g}_{J_C})$. Since $\deg(J_C) = \deg(I) \cdot \deg_x(h_{l,C}(\bar{x}_l, x))$, the lemma statement is equivalent to showing,

$$\deg_x(\tilde{g}_I) \ \geq \ \sum_{C \in \mathrm{des}(N)} \deg_x(h_{l,C}(\bar{x}_l, x)) \cdot \deg_x(\tilde{g}_{J_C}) \ . \tag{3}$$

Continuing with the notation of a particular child C, fix an $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I)$. Since J_C is a split ideal, $h_{l,C}(\bar{a}, x)$ (of degree d'_C) can be written as $\prod_{i=1}^{d'_C}(x - c_i)$, where each $c_i \in \mathbb{F}_p$ and are distinct. Then, each c_i is also a root of $\tilde{g}_I(\bar{a}, x)$, say with multiplicity $m_i \in \mathbb{N}$. So, there exists $G(x) \in \mathbb{F}_p[x]$ (coprime to $x - c_i$), such that, $\tilde{g}_I(\bar{a}, x) \equiv (x - c_i)^{m_i} \cdot G(x) \bmod p$. Lifting this equation mod p^k, there exists $G_1(x) \in R[x]$, of degree less than m_i, and a unique lift $G_2(x) \in R[x]$ of $G(x)$ (Hensel lemma (21)) : $g_I(\bar{a}, x) \equiv ((x - c_i)^{m_i} + pG_1(x)) \cdot G_2(x) \bmod p^k$. Substituting $x \to c_i + px$, we get, $g_I(\bar{a}, c_i + px) \equiv ((px)^{m_i} + pG_1(c_i + px)) \cdot G_2(c_i + px) \bmod p^k$.

Let $\bar{b}_i = (\bar{a}, c_i) \in \mathcal{Z}_{\mathbb{F}_p}(J_C)$. We know that $\tilde{g}_{J_C}(\bar{b}_i, x) = f_I(\bar{a}, c_i + px)/p^{v'} \bmod p$ is nontrivial. This implies that, $((px)^{m_i} + pG_1(c_i + px))/p^{v'} \bmod p$ is a nonzero polynomial of degree at most m_i ($\because p \nmid G_2(c_i)$).

Since $G_2(c_i + px) \not\equiv 0 \bmod p$ is a unit, $\deg_x(\tilde{g}_{J_C}(\bar{b}_i, x)) = \deg_x(\tilde{g}_{J_C}) \leq m_i$ (Eqn. 2). Summing up over all the roots c_i of $\tilde{g}_I(\bar{a}, x)$,

$$\sum_{i=1}^{d'_C} \deg_x(\tilde{g}_{J_C}(\bar{b}_i, x)) \;=\; d'_C \cdot \deg_x(\tilde{g}_{J_C}) \;\leq\; \sum_{i=1}^{d'_C} m_i \;=: d_C(g_I)\;.$$

Summing over all children $C \in \mathrm{des}(N)$ (using Eqn. 1, factorization of h_{l+1} & distinctness of $\mathbb{F}_p$-roots), we deduce,

$$\sum_{C \in \mathrm{des}(N)} \deg_x(h_{l,C}) \deg_x(\tilde{g}_{J_C}) \;\leq\; \sum_C d_C(g_I) \;\leq\; \deg_x(\tilde{g}_I(\bar{a}, x)) \;=\; \deg_x(\tilde{g}_I)\;.$$

This proves Eqn. 3, and hence the lemma. □

Define the degree of list $\mathcal{L}$ as, $\deg(\mathcal{L}) := \Sigma_{I \in \mathcal{L}} \deg(I)$.

Lemma 17 (Bounding $|RT|$, $\deg(I)$, $\deg(\mathcal{L})$, $|\mathcal{L}|$)**.** *Let RT be the roots-tree constructed from the execution of Algorithm 1. The number of leaves of RT, resp. $\deg(\mathcal{L})$, is at most $d = \deg(f(x))$. Also, the size $|RT|$ of the roots-tree (hence, the number of iterations by Algorithm 1) is bounded by dk.*

Proof. Applying Lemma 16 inductively, sum of the degrees of nodes at any level is bounded by the degree d of the root node. In particular,

1) We can extend every leaf to bring it to the last level (create a chain of nodes of same degree) without changing the degree distribution property. So, $\deg(\mathcal{L}) = \Sigma_{I \in \mathcal{L}} \deg(I) \leq d$. Since the number of leaves is $\geq |\mathcal{L}|$, we get $|\mathcal{L}| \leq d$.

2) For any split ideal I in stack S, $\deg I \leq d$.

3) Since the depth of the roots-tree is at most k, $|RT| \leq kd$. □

Lemma 18 (Computation cost at a node)**.** *Computation cost at each node of RT (time taken by Algorithm 1 in every iteration of the while loop) is bounded by poly$(d, k \log p)$.*

Proof. During an iteration, the major computations performed by the algorithm are— testing for zerodivisors (Step 9), computing modular gcd (Step 13), computing reduced f_I (Steps 3, 18), performing reduction for repeated squaring (Step 12), and factoring ideals (Step 20).

These operations are described by Lemmas 28, 29, 30, 32 and 33. All of them take time poly$(d, k \log p, \deg(I))$, where I is the concerned triangular ideal.

For any split ideal I (or its lift $\hat{I}$), we know that $\deg(I) \leq d$ (Lemma 17). So, Steps 3, 9, 13, 18, 20 take time poly$(d, k \log p)$. Step 12 to compute repeated squaring modulo $I + \langle \tilde{g} \rangle$ takes time poly$(\deg_x(\tilde{g}), \deg(I), k \log p)$ (using Lemma 28). Since I is a split ideal with $\deg(I) \leq d$, and degree of $\tilde{g}$ is at most d, so Step 12 also takes poly$(d, k \log p)$ time.

Hence the computation cost at each node is poly$(d, k \log p)$. □

Proof of Theorem 1. The definition of roots-tree shows that the number of leaves upper bound the number of all maximal split ideals in $\mathcal{L}$. Lemmas 17 and 18 show that the time complexity of Algorithm 1 is bounded by $\text{poly}(d, k \log p)$ (by bounding both number of iterations and the cost of computation at each iteration). Using Lemma 8 on the output of Algorithm 1, we get the exact count on the number of roots of $f \bmod p^k$ in time $\text{poly}(d, k \log p)$. □

4 Proof of Theorem 2

A polynomial f can be factored mod p^k if it has two basic-irreducible factors of different degree (using distinct degree factorization [vzGP01] and Hensel Lemma 21).

If two basic-irreducible factors appear with different exponents/multiplicities, then again f can be factored (using formal derivatives [vzGP01] and Hensel Lemma 21).

So, for factoring $f \bmod p^k$, we can assume $f \equiv (\varphi_1 \dots \varphi_t)^e + ph \bmod p^k$, where every $\varphi_i \in (\mathbb{Z}/\langle p^k \rangle)[x]$ is a basic-irreducible polynomial of a fixed degree b. Also, $d := \deg(f) = bte$. Let us fix this assumption for this section, unless stated explicitly.

A basic-irreducible factor of $f \bmod p^k$ has the form $\varphi_i + pw_i(x) \bmod p^k$, for $i \in [t]$ (Lemma 22).

If $b = 1$, counting basic-irreducible factors of f is equivalent to counting roots of f.

When $b > 1$, we prove a simple generalization of this idea; it is enough to count all the roots of f in the ring extension $\mathbb{Z}[y]/\langle p^k, \varphi(y) \rangle$, where $\varphi(y)$ is an irreducible mod p of degree-b. These rings are called *Galois rings*, we denote them by $G(p^k, b)$ (unique, for fixed k and b, up to isomorphism).

4.1 Reduction to root-counting in $G(p^k, b)$

By Lemma 22, any basic-irreducible factor of $f \bmod p^k$ is a factor of a unique $({\varphi_i}^e + pw_i(x))$; and φ_i are coprime mod p. So in this subsection, for simplicity of exposition, we will assume that $f(x)$ equals $\varphi^e \bmod p$ (φ is a monic degree-b irreducible mod p).

Define $G := G(p^k, b)$. Let $y_0, y_1, \dots, y_{b-1}$ be the roots of $\varphi(x)$ in G (Claim 24). Wlog, take $y := y_0$, $y_i \equiv y^{p^i} \bmod p$, for all $i \in \{0, \dots, b-1\}$ (Frobenius conjugates in $\mathbb{F}_p$). Note that $G \cong (\mathbb{Z}/\langle p^k \rangle)[y] =: G'$. We will prefer to use G' below.

The lemma below associates a root of f, in G or G', to a unique basic-irreducible factor of f in $(\mathbb{Z}/\langle p^k \rangle)[x]$.

Lemma 19 (Root to factor)**.** *Let $r(y) \in G'$ be a root of $f(x)$. Then, $h(x) := \prod_{i=0}^{b-1}(x - r(y_i))$ is the unique basic-irreducible factor of f having root $r(y)$. We say: $h(x)$ is the basic-irreducible factor* associated to root $r(y)$.

Proof. The coefficients of h are symmetric polynomials in $r(y_i)$ (over $0 \le i < b$). Since the automorphism $\psi_1 : y \to y_1$ of G' (as defined in Claim 25) permutes $r(y_i)$'s ($\because$ it permutes y_i's), it fixes all the coefficients of h. From Claim 25, all these coefficients are then in $\mathbb{Z}/\langle p^k \rangle$. Hence, $h \in (\mathbb{Z}/\langle p^k \rangle)[x]$.

If $r(y)$ is a root of another polynomial h' in $(\mathbb{Z}/\langle p^k \rangle)[x]$, then $r(y_i)$'s are also roots of h' (applying automorphisms ψ_i of G'). Since these roots are coprime mod p, we actually get: $h | h'$. Thus, h is the unique monic irreducible factor of f containing $r(y)$.

Looking mod p, $r(y_i)$'s are a permutation of the roots of $\varphi(x)$, so $h(x) \equiv \varphi(x) \bmod p$. Hence, $h(x)$ is the unique monic basic-irreducible factor of f having root $r(y)$. □

Following is the reduction to counting all roots of f in G.

Theorem 20 (Factor to root)**.** *Any degree-b basic-irreducible factor of $f \bmod p^k$ has exactly b roots in G. Conversely, if f has a root $r(y) \in G$, then it must be a root of a unique degree-b basic-irreducible factor of $f \bmod p^k$.*

So, the number of degree-b basic-irreducible factors of $f \bmod p^k$ is exactly the number of roots, of f in G, divided by the degree b.

Proof. By Lemma 19 (& uniqueness of Galois rings), for every root $r(y) \in G$ of f, we can associate a unique basic-irreducible factor of $f(x)$.

Conversely, let $h(x) =: \varphi(x) + pw(x)$ be a basic-irreducible factor of $f(x)$. It splits completely in G (as, $h(x) \equiv \varphi \bmod p$; first factor in $G/\langle p\rangle$ and then Hensel lift to G). So, h has exactly b roots in G, each of them is also a root of f in G.

Hence the theorem statement follows. □

Remark. This 'irreducible factor vs root' correspondence, for $f \bmod p^k$, breaks down if G is *not* a Galois ring. E.g., what happens when the ring is $\mathbb{Z}[y]/\langle p^k, y^2 - p\rangle$?

4.2 Counting roots in $G(p^k, b)$– Wrapping up Thm. 2

In this section, we show how to count the roots of $f \equiv (\varphi_1\varphi_2 \dots \varphi_t)^e + ph(x) \bmod p^k$ in $G(p^k, b)$. Since $G := G(p^k, b)$ is a Galois ring, so $G/\langle p\rangle = \mathbb{F}_{p^b} =: \mathbb{F}_q$. (Recall: $R = \mathbb{Z}/\langle p^k\rangle$.)

Split ideals and zerosets in the Galois ring: First, we will modify the definition of zerosets (Section 2) to include zeros of f in G. A *G-zeroset* of $f(x) \in R[x]$ will be defined as $\mathcal{Z}_G(f) := \{r \in G \mid f(r) \equiv 0 \bmod p^k\}$. Similarly, for an ideal $I \subseteq \mathbb{F}_p[\bar{x}_l]$, its $\mathbb{F}_q$-zeroset is defined as $\mathcal{Z}_{\mathbb{F}_q}(I) := \{\bar{a} = (a_0, \dots, a_l) \in (\mathbb{F}_q)^{l+1} \mid g(\bar{a}) \equiv 0 \bmod p^k, \forall g \in I\}$.

The definition of triangular ideals, split ideals and maximal split ideals will remain exactly same (generators defined over $\mathbb{F}_p$, Section 2), except that in the third condition for split ideals, zeroset will be over $\mathbb{F}_q$ instead of $\mathbb{F}_p$. But, they can now be seen as storing potential roots of $f(x)$ in G (or, storing potential basic irreducible factors of $f \bmod p^k$). The reason is, a root $r(y) \in G$ of $f \bmod p^k$ can be viewed as, $r(y) = r_0(y) + pr_1(y) + p^2r_2(y) + \ldots + p^{k-1}r_{k-1}(y)$, where each $r_i(y) \in G/\langle p\rangle = \mathbb{F}_q$. So, the decomposition of formal variable $x =: x_0 + px_1 + p^2x_2 + \ldots + p^{k-1}x_{k-1}$, now represents candidates for r_0, r_1, and so on, over $\mathbb{F}_q$.

A split ideal $I_l \subseteq \mathbb{F}_p[\bar{x}_l]$, defined as $I_l := \langle h_0(x_0), \dots, h_l(\bar{x}_l)\rangle$, now implicitly stores the candidates for (r_0) in h_0, (r_0, r_1) in h_1, and so on. These, in turn, give candidates for basic-irreducible factors of $f \bmod p^{l'}$ (some $l' \le k$).

In particular, when I_l is a maximal split ideal, an $\bar{r}_l$ implicitly denote a basic-irreducible factor of $f \bmod p^k$. The number of such factors is $\deg(I_l) \cdot q^{k-l-1}/b$ (Theorem 20 & Lemma 8).

Split ideals follow all the properties given in Section 2, just by replacing the fact that roots belong to $\mathbb{F}_q$ and not $\mathbb{F}_p$.

Description of the modified algorithm: Algorithm 1, to count roots in R, extends directly to count roots in G. The algorithm is exactly same except one change: to compute GCD (Steps 3 and 13), we now use the Frobenius polynomial $x^q - x$ instead of the prior $x^p - x$ (GCD computation implicitly stores the candidate roots, they are in $\mathbb{F}_q$ now).

So the algorithm works as follows:

1. It gets $f(x) \equiv (\varphi_1 \dots \varphi_t)^e + pw(x) \bmod p^k$ as input, computes gcd $h_0(x) := \gcd(f(x), x^q - x)$ over $\mathbb{F}_p$. Since $x^q - x$, over $\mathbb{F}_p$, is the product of all irreducible factors of degree dividing b, we

deduce: $h_0(x) = \varphi_1 \dots \varphi_t \bmod p$; and define the first split ideal $I_0 := \langle h_0 \rangle$. (Note– We do not have access to φ_i's themselves.)

Remark. The length 1 split ideal stores all the roots of f in $G/\langle p \rangle$, or all the basic irreducible factors of $f \bmod p$; as $h_0(x) = \varphi_1 \dots \varphi_t$. Also, its degree is tb, which when divided by b, gives the count of the basic-irreducible factors of $f \bmod p$.

2. The algorithm then successively looks for the next precision candidates. It computes h_l by taking gcd with $x^q - x$, and adds it to the previous ideal I_{l-1} like before.

3. All the supporting algebraic algorithms and lemmas (given in appendix) work the same as before; since they are being passed the same parameters— a split ideal, or a triangular ideal, or a polynomial over R.

Thus, a similar proof of correctness and time complexity can be given as before.

Proof of Theorem 2. Consider a univariate $f(x) \bmod p^k$. As discussed in the beginning of this section, $f \bmod p^k$ can be efficiently factorized as $f \equiv \prod_{i=1}^{m} f_i \bmod p^k$, where each $f_i(x)$ is a power of a product of degree-b_i irreducible polynomials mod p (i.e. of the form $\equiv (\varphi_1 \varphi_2 \dots \varphi_t)^e + ph(x)$, where φ_j is a degree-b_i irreducible mod p).

On each such $f_i \bmod p^k$, we use Algorithm 1 with the new Frobenius polynomial $(x^{q_i} - x)$ $(q_i = p^{b_i})$, in Steps 3 and 18, as discussed above. Let the final list output, for $f_i \bmod p^k$, be $\mathcal{L}_i =: \{I_1(l_1, D_1), \dots, I_n(l_n, D_n)\}$. Thus, we get the count on the $G(p^k, b_i)$-roots of $f_i \bmod p^k$ as $\Sigma_{j=1}^{n} D_j q_i^{k-l_j}$ (Lemma 8). Using Theorem 20, the number of the degree-b_i basic-irreducible factors of $f \bmod p^k$ is $B_k(f_i) := (1/b_i) \times \Sigma_{j=1}^{n} D_j q_i^{k-l_j}$.

Using Lemma 22, we get the count on the basic-irreducible factors of $f \bmod p^k$ as, $B_k(f) = \Sigma_{i=1}^{m} B_k(f_i)$.

For the time complexity, only difference is the repeated-squaring to compute the reduced form of polynomial $x^{q_i} - x$ (Steps 3, 12), it will take $b_i \log p$ operations instead of $\log p$ operations. But $b_i \leq d$, so the algorithm runs in time $\text{poly}(d, k \log p)$ (& remains deterministic).

□

5 Conclusion

There are well known efficient deterministic algorithms to count the number of roots/irreducible factors over prime characteristic. Surprisingly, not many results are known when the characteristic is a *prime-power*. The main difficulty is that the ring has *non*-unique factorization.

We give the first efficient deterministic algorithm to count the number of basic-irreducible factors modulo a prime-power. Restricting it to degree-one irreducibles, we get a deterministic polynomial-time algorithm to count the roots too. This is achieved by storing and improving roots (wrt precision) virtually using *split ideals* (we do not have access to roots directly). As a corollary: we can compute the Igusa zeta function deterministically, and we also get a deterministic algorithm to count roots in p-adic rings (resp. formal power-series ring).

Many interesting questions still remain to be tackled. For p-adic fields, there is only a randomized method to count the number of irreducible factors. Analogously, the question of counting irreducible factors modulo a prime-power also remains open; no efficient method is known even in the randomized

setting. The *ramified* roots seem to elude practical methods. On the other hand, the problem of actually *finding* an irreducible factor (resp. a root) deterministically, seems much harder; it subsumes the analogous classic problem in prime characteristic.

Acknowledgements. We thank Vishwas Bhargava for introducing us to the open problem of factoring $f \bmod p^3$ and the related prime-power questions. A.D. thanks Sumanta Ghosh for the discussions. N.S. thanks the funding support from DST (DST/SJF/MSA-01/2013-14). R.M. would like to thank support from DST through grant DST/INSPIRE/04/2014/001799. We thank anonymous reviewers for their helpful comments and suggestions to improve the introduction section of this paper.

References

[Apo13] Tom M Apostol. *Introduction to analytic number theory*. Springer Science & Business Media, 2013. 2

[Bha97] Manjul Bhargava. P-orderings and polynomial functions on arbitrary subsets of dedekind rings. *Journal fur die Reine und Angewandte Mathematik*, 490:101–128, 1997. 2

[BLQ13] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin. Polynomial root finding over local rings and application to error correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 24(6):413–443, 2013. https://link.springer.com/article/10.1007/s00200-013-0200-5. 2, 3, 6, 7

[CG00] David G Cantor and Daniel M Gordon. Factoring polynomials over p-adic fields. In *International Algorithmic Number Theory Symposium*, pages 185–208. Springer, 2000. 2, 3

[CGRW18] Qi Cheng, Shuhong Gao, J Maurice Rojas, and Daqing Wan. Counting roots of polynomials over prime power rings. In *Thirteenth Algorithmic Number Theory Symposium, ANTS-XIII*. Mathematical Sciences Publishers, 2018. arXiv:1711.01355. 2, 3, 4, 6, 7

[Chi87] AL Chistov. Efficient factorization of polynomials over local fields. *Dokl. Akad. Nauk SSSR*, 293(5):1073–1077, 1987. 2, 3

[Chi94] AL Chistov. Algorithm of polynomial complexity for factoring polynomials over local fields. *Journal of mathematical sciences*, 70(4):1912–1933, 1994. 2, 3

[CP56] M Chojnacka-Pniewska. Sur les congruences aux racines données. In *Annales Polonici Mathematici*, volume 3, pages 9–12. Instytut Matematyczny Polskiej Akademii Nauk, 1956. 2

[CZ81] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981. 2, 23

[Den91] Jan Denef. Report on Igusa's local zeta function. *Astérisque*, 730-744(201-203):359–386, 1991. 2, 3

[DH01] Jan Denef and Kathleen Hoornaert. Newton polyhedra and Igusa's local zeta function. *Journal of number Theory*, 89(1):31–64, 2001. 2

[DM97] Bruce Dearden and Jerry Metzger. Roots of polynomials modulo prime powers. *European Journal of Combinatorics*, 18(6):601–606, 1997. 2

[DMS19] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. Efficiently factoring polynomials modulo p^4. *The 44th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, 2019. https://www.cse.iitk.ac.in/users/nitin/papers/factor-mod-p4.pdf. 1, 2, 6

[EK90] Andrzej Ehrenfeucht and Marek Karpinski. *The computational complexity of (xor, and)-counting problems*. International Computer Science Inst., 1990. 3

[Hen18] Kurt Hensel. Eine neue theorie der algebraischen zahlen. *Mathematische Zeitschrift*, 2(3):433–452, Sep 1918. 22

[Igu74] Jun-ichi Igusa. Complex powers and asymptotic expansions. i. functions of certain types. *Journal für die reine und angewandte Mathematik*, 268:110–130, 1974. 2, 3

[Kli97] Adam Klivans. Factoring polynomials modulo composites. Technical report, Carnegie-Mellon Univ, Pittsburgh PA, Dept of CS, 1997. 1

[Kob77] Neal Koblitz. P-adic numbers. In *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, pages 1–20. Springer, 1977. 3

[KRRZ19] Leann Kopp, Natalie Randall, Joseph Rojas, and Yuyu Zhu. Randomized polynomial-time root counting in prime power rings. *Mathematics of Computation*, 2019. 2, 3, 6, 7

[Lau04] Alan GB Lauder. Counting solutions to equations in many variables over finite fields. *Foundations of Computational Mathematics*, 4(3):221–267, 2004. 2

[LN94] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge university press, 1994. 23

[Mau01] Davesh Maulik. Root sets of polynomials modulo prime powers. *Journal of Combinatorial Theory, Series A*, 93(1):125–140, 2001. 2

[McD74] Bernard R McDonald. *Finite rings with identity*, volume 28. Marcel Dekker Incorporated, 1974. 23

[NRS17] Vincent Neiger, Johan Rosenkilde, and Éric Schost. Fast computation of the roots of polynomials over the ring of power series. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 349–356. ACM, 2017. 2

[NZM13] Ivan Niven, Herbert S Zuckerman, and Hugh L Montgomery. *An introduction to the theory of numbers*. John Wiley & Sons, 2013. 2

[Săl05] Ana Sălăgean. Factoring polynomials over $\mathbb{Z}_4$ and over certain galois rings. *Finite fields and their applications*, 11(1):56–70, 2005. 1, 2

[Sha93] Adi Shamir. On the generation of multivariate polynomials which are hard to factor. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 796–804. ACM, 1993. 1

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009. 26, 29

[Sie55] Wacław Sierpiński. Remarques sur les racines d'une congruence. *Annales Polonici Mathematici*, 1(1):89–90, 1955. 2

[Sir17] Carlo Sircana. Factorization of polynomials over $\mathbb{Z}/(p^n)$. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 405–412. ACM, 2017. 1, 2

[vzGH96] Joachim von zur Gathen and Silke Hartlieb. Factorization of polynomials modulo small prime powers. Technical report, Paderborn Univ, 1996. 1, 2

[vzGH98] Joachim von zur Gathen and Silke Hartlieb. Factoring modular polynomials. *Journal of Symbolic Computation*, 26(5):583–606, 1998. (Conference version in ISSAC'96). 1, 2, 3

[vzGP01] Joachim von zur Gathen and Daniel Panario. Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation*, 31(1-2):3–17, 2001. 17

[Zas69] Hans Zassenhaus. On hensel factorization, I. *Journal of Number Theory*, 1(3):291–311, 1969. 22

[ZG03] WA Zuniga-Galindo. Computing Igusa's local zeta functions of univariate polynomials, and linear feedback shift registers. *Journal of Integer Sequences*, 6(2):3, 2003. 2

A Preliminaries

Lifting factorization: Below we state a lemma, originally due to Kurt Hensel [Hen18], for $\mathcal{I}$-adic lifting of factorization of a given univariate polynomial. Over the years, Hensel's lemma has acquired many forms in different texts, version presented here is due to Zassenhaus [Zas69].

Lemma 21 (Hensel's lemma [Hen18])**.** *Let R be a commutative ring with unity, denote the polynomial ring over it by $R[x]$. Let $\mathcal{I} \subseteq R$ be an ideal of ring R. Given a polynomial $f(x) \in R[x]$, suppose f factorizes as*

$$f = gh \bmod \mathcal{I},$$

such that $gu + hv = 1 \bmod \mathcal{I}$ (for some $g, h, u, v \in R[x]$). Then, given any $l \in \mathbb{N}$, we can efficiently compute $g^*, h^*, u^*, v^* \in R[x]$, such that,

$$f = g^* h^* \bmod \mathcal{I}^l.$$

Here $g^* = g \bmod \mathcal{I}$, $h^* = h \bmod \mathcal{I}$ and $g^*u^* + h^*v^* = 1 \bmod \mathcal{I}^l$ (i.e. pseudo-*coprime lifts). Moreover g^* and h^* are unique up to multiplication by a unit.*

Using Hensel's lemma, for the purpose of counting roots (resp. basic-irreducible factors), a univariate polynomial $f(x) \in \mathbb{Z}[x]$ can be assumed to be a power of an irreducible modulo p.

Lemma 22. *By the fundamental theorem of algebra, a univariate $f(x) \in \mathbb{Z}[x]$ factors uniquely, over $\mathbb{F}_p$, into coprime powers as, $f \equiv \prod_{i=1}^{m} {\varphi_i}^{e_i}$, where each $\varphi_i \in \mathbb{Z}[x]$ is irreducible mod p and $m, e_i \in \mathbb{N}$. Then, for all $k \in \mathbb{N}$,*

1. *f factorizes mod p^k as $f = g_1 g_2 \dots g_m$, where g_i's are mutually co-prime mod p^k and $g_i \equiv {\varphi_i}^{e_i} \bmod p$, for all $i \in [m]$.*
2. *any basic-irreducible factor of $f(x) \bmod p^k$ is a basic-irreducible factor of a unique $g_j \bmod p^k$, for some $j \in [m]$. Let $B_k(h)$ denote the number of (coprime) basic-irreducible factors of $h(x) \bmod p^k$. Then, $B_k(f) = \Sigma_{i=1}^{m} B_k(g_i)$.*
3. *any root of f mod p^k is a root of a unique g_i mod p^k. Let $N_k(h)$ denote the number of (distinct) roots of $h(x) \bmod p^k$. Then, $N_k(f) = \Sigma_{i=1}^{m} N_k(g_i)$.*

Proof. We can apply Hensel's lemma by taking ring $R := \mathbb{Z}$ and ideal $\mathcal{I} := \langle p \rangle$. The co-prime factorization of $f \bmod p$ lifts to a unique coprime factorization $f \equiv g_1 g_2 \dots g_m \bmod p^k$, for any $k \in \mathbb{N}$ and $g_i \equiv {\varphi_i}^{e_i} \bmod p$.

Any basic-irreducible factor $h(x)$ of $f(x) \bmod p^k$ has to be $h \equiv \varphi_i \bmod p$ for some $i \in [m]$; otherwise, h will become reducible mod p. Since g_i's are co-prime and $h | f \bmod p^k$, h must divide a unique g_i. So, any basic-irreducible factor h of $f(x) \bmod p^k$ is a basic-irreducible factor of a unique $g_j \bmod p^k$. Clearly, any basic-irreducible factor of a g_i is also a basic-irreducible factor of $f \bmod p^k$. This proves $B_k(f) = \Sigma_{i=1}^{m} B_k(g_i)$.

The third part follows from a similar reasoning as the second part. □

Root finding over a finite field: The following theorem, called CZ in this paper and given by Cantor-Zassenhaus [CZ81], finds all roots of a given univariate polynomial over a finite field in randomized polynomial time. (Equivalently, it finds all irreducible factors as well.)

Theorem 23 (Cantor-Zassenhaus Algo (CZ))**.** *Given a univariate degree d polynomial $f(x)$ over a finite field $\mathbb{F}_q$, all roots of f in $\mathbb{F}_q$ can be found in randomized poly$(d, \log q)$ time.*

A.1 Properties of Galois rings– Analogues of finite fields

A *Galois ring*, of characteristic p^k and size p^{kb}, is denoted by $G(p^k, b)$ (where p is a prime, $k, b \in \mathbb{N}$). It is known that two Galois rings of same characteristic and size are isomorphic to each other. We will define Galois ring $G(p^k, b)$ as the ring $\mathbb{G} := \mathbb{Z}[y]/\langle p^k, \varphi(y) \rangle$, where $\varphi(y) \in \mathbb{Z}[y]$ is an irreducible mod p of degree b [McD74]. Let us prove some useful properties of $\mathbb{G}$ below.

Claim 24 (Roots of φ)**.** *Let $\varphi'(x) \in \mathbb{Z}[x]$ be any irreducible mod p of degree b. There are b distinct roots of $\varphi'(x)$ in $\mathbb{G}$. Let r denote one of the roots, then all other roots, modulo p, are of the form r^{p^i} ($i \in \{0, \dots, b-1\}$).*

Proof. $\mathbb{G}/\langle p \rangle$ is isomorphic to the finite field of degree b over $\mathbb{F}_p$. So, irreducible $\varphi'(x) \in \mathbb{F}_p[x]$ has exactly b roots in $\mathbb{G}/\langle p \rangle$ [LN94, Ch.2]. By Hensel Lemma 21, roots in $\mathbb{G}/\langle p \rangle$ can be lifted to $\mathbb{G}$ uniquely. Hence, $\varphi'(x)$ has exactly b distinct roots in $\mathbb{G}$. Modulo p, they are of the form r^{p^i} ($i \in \{0, \dots, b-1\}$) for a root r (lifted from roots in $\mathbb{G}/\langle p \rangle$). □

Using Claim 24, denote roots of $\varphi(x)$ as $y_0, \dots, y_{b-1}$; here $y_i \equiv y_0^{p^i} \bmod p$ for all $i \in \{0, \dots, b-1\}$. For all roots y_j, $\mathbb{G} \equiv R[y_j]$. In other words, y_j generate the extension $\mathbb{G}$ over R.

Claim 25 (Symmetries of $\mathbb{G}$)**.** *There are exactly b automorphisms of $\mathbb{G}$ fixing $R = \mathbb{Z}/\langle p^k \rangle$, denoted by ψ_j ($j \in \{0, \ldots, b-1\}$). Each of these automorphisms can be described by a map taking y_0 to one of the roots of $\varphi(x)$ and fixing R. Wlog, assume ψ_j maps $y_0 \to y_j$.*

Moreover, for all j coprime to b, ψ_j fixes R and nothing else.

Proof. Since coefficients of $\varphi(x)$ belong to R, an automorphism fixing R should map the root y_0 to another of its roots y_j. We only need to show that ψ_j is an automorphism (it is a valid map because $y_j \in \mathbb{G}$)

Writing elements of $\mathbb{G}$ in terms of y_0 (i.e. $\mathbb{G} \cong R[y_0]$), it can be verified that $\psi_j(ab) = \psi_j(a)\psi_j(b)$ and $\psi_j(a+b) = \psi_j(a) + \psi_j(b)$, so ψ_j is a homomorphism.

Similarly, if $\psi_j(g) = 0$, writing g in terms of y_0, we get that $g = 0$. So, kernel of ψ_j is the set $\{0\}$; thus, it is an isomorphism.

For the moreover part, let ψ_j be such that j is coprime to b. We will show a stronger statement by induction: for any $i \leq k-1$, if $a(y_0) = \psi_j(a(y_0))$ in $\mathbb{G}/\langle p^i \rangle$, then $a(y_0) \in \mathbb{Z}/\langle p^i \rangle$.

Base case: If $i = 1$ and $j = 1$, then $a(y_0) = \psi_1(a(y_0)) \bmod p \Rightarrow a(y_0) = a(y_0)^p \bmod p$. It means $a(y_0) \in \mathbb{Z}/\langle p \rangle$.

If j is coprime to b, then ψ_j generates ψ_1 modulo p. So, $a(y_0) = \psi_j(a(y_0)) \bmod p$ implies that, $a(y_0) \bmod p =: a_0 \in \mathbb{Z}/\langle p \rangle$.

This argument also proves: for any $i \leq k$, if $a(y_0) = a(y_j)$ in $\mathbb{G}/\langle p^i \rangle$, then $a(y_0) \in \mathbb{F}_p$ (in other words, $a(y_0)$ is y_0 free).

Induction step: Let us assume that $a(y_0) = \psi_j(a(y_0))$ in $\mathbb{G}/\langle p^i \rangle$. By the previous argument, $a(y_0) = a_0 + pa'(y_0)$, where $a_0 \in \mathbb{Z}/\langle p \rangle$ and $a'(y_0) \in \mathbb{G}/\langle p^{i-1} \rangle$.

From the definition, $a(y_0) = \psi_j(a(y_0))$ iff $a'(y_0) = \psi_j(a'(y_0))$ in $\mathbb{G}/\langle p^{i-1} \rangle$. By induction hypothesis, the latter is equivalent to $a'(y_0) \in \mathbb{Z}/\langle p^{i-1} \rangle$. So, $a(y_0) \in \mathbb{Z}/\langle p^i \rangle$.

Hence, the only fixed elements under the map ψ_j (j coprime to b) are integers; in $\mathbb{Z}/\langle p^k \rangle$. □

B Proofs of Section 2

Proof of Lemma 5. It is enough to show the lemma for $j = l-1$. It is easy to observe that I_{l-1} is triangular.

Looking at the second condition for being a split ideal, $|\mathcal{Z}_{\mathbb{F}_p}(I_{l-1})| \leq \prod_{i=0}^{l-1} \deg_{x_i}(h_i)$ follows because a degree $d \geq 1$ polynomial can have at most d roots in $\mathbb{F}_p$.

To show equality, notice that for any $\bar{a} = (a_0, \ldots, a_{l-1}) \in \mathcal{Z}_{\mathbb{F}_p}(I_{l-1})$, $\deg_{x_l}(h_l(\bar{a}, x_l))$ is bounded by $\deg_{x_l}(h_l)$. This implies $h_l(\bar{a}, x_l)$ can have at most $\deg_{x_l}(h_l)$ roots in $\mathbb{F}_p$. If $|\mathcal{Z}_{\mathbb{F}_p}(I_{l-1})| < \prod_{i=0}^{l-1} \deg_{x_i}(h_i)$ then $|\mathcal{Z}_{\mathbb{F}_p}(I_l)| < \deg_{x_l}(h_l) \cdot \prod_{i=0}^{l-1} \deg_{x_i}(h_i)$, contradicting that I_l is a split ideal. [3]

For the third condition, since I_l is a split ideal, for any $(a_0, \ldots, a_{l-1}) \in \mathcal{Z}_{\mathbb{F}_p}(I_{l-1})$, $f(a_0 + pa_1 + \ldots + p^l a_l) \equiv 0 \bmod p^{l+1} \Rightarrow f(a_0 + pa_1 + \ldots + p^{l-1}a_{l-1}) \equiv 0 \bmod p^l$. □

Lemma 6 shows that a split ideal I can be decomposed in terms of ideals $I_{\bar{a}} := \langle x_0 - a_0, \ldots, x_l - a_l \rangle$, where $\bar{a} =: (a_0, \ldots, a_l)$ is a root of I. Before we prove this structural lemma, let us see some properties of these ideals $I_{\bar{a}}$'s.

Claim 26. *Let I be a split ideal.*

1. *For any ideal $I_{\bar{a}}$, quotient $\mathbb{F}_p[x_0, \ldots, x_l]/I_{\bar{a}} \cong \mathbb{F}_p$ is a* field.

[3] This argument also shows that every $\mathbb{F}_p$-zero of I_{l-1} 'extends' to exactly $\deg_{x_l}(h_l)$ many $\mathbb{F}_p$-zeros of I_l.

2. $I_{\bar{a}}$ *and* $I_{\bar{b}}$ *are* coprime *for any two distinct roots* $\bar{a}, \bar{b} \in \mathcal{Z}_{\mathbb{F}_p}(I)$. *This is because there exists* i, *for which* $a_i \neq b_i$; *yielding* $(a_i - b_i)^{-1}((x_i - b_i) - (x_i - a_i)) = 1$ *in the sum-ideal* $I_{\bar{a}} + I_{\bar{b}}$.

3. $I_{\bar{a}} \cap I_{\bar{b}} = I_{\bar{a}} I_{\bar{b}}$ *for any two distinct roots* $\bar{a}, \bar{b} \in \mathcal{Z}_{\mathbb{F}_p}(I)$. *It follows because there exist* $r_{\bar{a}} \in I_{\bar{a}}$ *and* $r_{\bar{b}} \in I_{\bar{b}}$, *s.t.,* $r_{\bar{a}} + r_{\bar{b}} = 1$. *So,* $r \in I_{\bar{a}} \cap I_{\bar{b}} \Rightarrow r = r(r_{\bar{a}} + r_{\bar{b}}) \in I_{\bar{a}} I_{\bar{b}}$. *On the other hand,* $I_{\bar{a}} I_{\bar{b}} \subseteq I_{\bar{a}} \cap I_{\bar{b}}$ *follows from the definition of the product-ideal.*

4. *Generalizing the previous point— for a set* A *of distinct roots* $\bar{a}$*'s,* $\bigcap_{\bar{a} \in A} I_{\bar{a}} = \prod_{\bar{a} \in A} I_{\bar{a}}$.

Proof of Lemma 6. We will prove this decomposition by applying induction on the length of the split ideal. For the base case, length of I is 1 and $I = \langle h_0(\bar{x}_0) \rangle \subseteq \mathbb{F}_p[x_0]$. Since I is a split ideal, $h_0(x_0) = \prod_{i=1}^{\deg(h_0)} (x_0 - a_i)$ for *distinct* $a_i \in \mathbb{F}_p$. So, $I = \prod_{i=1}^{\deg(h_0)} I_{a_i} = \bigcap_{i=1}^{\deg(h_0)} I_{a_i}$ by Claim 26.

Let I be a split ideal of length $l+1$, $I =: \langle h_0(\bar{x}_0), \ldots, h_l(\bar{x}_l) \rangle \subseteq \mathbb{F}_p[x_0, \ldots, x_l]$. Define ideal $I' := \langle h_0(\bar{x}_0), \ldots, h_{l-1}(\bar{x}_{l-1}) \rangle$. By Lemma 5, I' is a split ideal. From the induction hypothesis (& Claim 26), we have $I' = \bigcap_{\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I)} I'_{\bar{a}} = \prod_{\bar{a}} I'_{\bar{a}}$, where $I'_{\bar{a}} := \langle x_0 - a_0, \ldots, x_{l-1} - a_{l-1} \rangle$ for a zero $\bar{a} =: (a_0, \ldots, a_{l-1})$ of I'. We know that,

$$I \;=\; I' + \langle h_l(\bar{x}_l) \rangle \;=\; \prod_{\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I')} \left(I'_{\bar{a}} + \langle h_l(\bar{x}_l) \rangle \right) . \tag{4}$$

Claim 10 shows $\deg(h_l(\bar{a}, x_l)) = \deg_{x_l}(h_l)$ for all $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I')$, and $h_l(\bar{a}, x_l)$ splits completely over $\mathbb{F}_p$. So, for any $\bar{a} \in \mathcal{Z}_{\mathbb{F}_p}(I')$, $I'_{\bar{a}} + \langle h_l(\bar{x}_l) \rangle = \prod_{i=1}^{\deg_{x_l}(h_l)} I_{\bar{a}, b_i}$, where $(\bar{a}, b_i)$ are roots of I extended from $\bar{a}$. From Eqn. 4 (& Claim 26), $I = \prod_{\bar{b} \in \mathcal{Z}_{\mathbb{F}_p}(I)} I_{\bar{b}} = \bigcap_{\bar{b} \in \mathcal{Z}_{\mathbb{F}_p}(I)} I_{\bar{b}}$.

This finishes the inductive proof, completely factoring I. □

Lemma 8 shows that a root of a maximal split ideal represents a set of roots of $f \bmod p^k$ and provides the size of that set.

Proof of Lemma 8. By definition of a maximal split ideal, for any $\bar{a} = (a_0, \ldots, a_l) \in \mathcal{Z}_{\mathbb{F}_p}(I)$, $p^k | g(x)$ where $g(x) = f(a_0 + pa_1 + p^2 a_2 + \ldots + p^l a_l + p^{l+1} x)$. So, $g(x) = 0 \bmod p^k$ for any p^{k-l-1} choices of x. For each such fixing of x, $a_0 + pa_1 + p^2 a_2 + \ldots + p^l a_l + p^{l+1} x$ is a *distinct* root of $f(x) \bmod p^k$. Hence proved. □

C Computation modulo a triangular ideal– Reduce & Divide

For completeness, we show that it is efficient to reduce a polynomial $a(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]$ modulo a triangular ideal $J_l = \langle b_0(\bar{x}_0), b_1(\bar{x}_1), \ldots, b_l(\bar{x}_l) \rangle \subseteq \mathbb{G}[\bar{x}_l]$, where $\mathbb{G}$ is any Galois ring (in particular, $R = \mathbb{Z}/p^k$, or $\mathbb{F}_p$).

Note: J_l need not be a split ideal for $f \bmod p^k$, though the algorithms of this section work for split ideals ($\because$ they are triangular by definition).

Assumptions: In the generators of the triangular ideal we assume $\deg_{x_i} b_i(\bar{x}_i) \geq 2$ (for $0 \leq i \leq l$). Otherwise, we could eliminate variable x_i and work with fewer variables (& smaller length triangular ideal). Additionally, each $b_i(\bar{x}_i)$ (for $0 \leq i \leq l$) is monic (leading coefficient is 1 wrt x_i), and presented in a *reduced* form modulo the prior triangular ideal $J_{i-1} := \langle b_0(\bar{x}_0), \ldots, b_{i-1}(\bar{x}_{i-1}) \rangle \subseteq \mathbb{G}[\bar{x}_{i-1}]$.

Let us first define reduction mod an ideal (assume $\mathbb{G}$ to be the Galois ring $G(p^k, b)$).

Definition 27 (Reduction by a triangular ideal)**.** *The* reduction *of a multivariate polynomial* $a(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]$ *by a triangular ideal* $J_l = \langle b_0(\bar{x}_0), \ldots, b_l(\bar{x}_l)\rangle \subseteq \mathbb{G}[\bar{x}_l]$ *is the unique polynomial* $\tilde{a}(\bar{x}_l) \equiv a(\bar{x}_l) \bmod J_l$*, where* $\deg_{x_i}(\tilde{a}) < \deg_{x_i}(b_i)$*, for all* $i \in \{0, \ldots, l\}$*.*

Idea of reduction: The idea behind the algorithm is inspired from the univariate reduction. If $l = 0$, then reduction of $a(x_0)$ modulo $b_0(x_0)$ is simply the remainder of the division of a by b_0 in the underlying polynomial ring $\mathbb{G}[x_0]$. For a larger l, the reduction of $a(\bar{x}_l)$ modulo the triangular ideal $J_l = \langle b_0(x_0), \ldots, b_l(\bar{x}_l)\rangle$ is the remainder of the division of $a(\bar{x}_l)$ by $b_l(\bar{x}_l)$ in the polynomial ring $(\mathbb{G}[x_0, \ldots, x_{l-1}]/J_{l-1})[x_l]$. The fact that b_l is monic, helps in generalizing 'long division'.

Input: An $a(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]$ and a triangular ideal $J_l = \langle b_0(\bar{x}_0), \ldots, b_l(\bar{x}_l)\rangle \subseteq \mathbb{G}[\bar{x}_l]$.

Output: Reduction $\tilde{a}$ of a mod J_l as defined above.

Algorithm 2 Reduce $a(\bar{x}_l)$ modulo J_l

1: **procedure** REDUCE($a(\bar{x}_l)$, J_l)
2: **if** $l = 0$ **then**
3: [**Reduce** $a(x_0)$ **by** $b_0(x_0)$] return remainder of univariate division of a by b_0 in $R[x_0]$.
4: **end if**
5: $d_a \leftarrow \deg_{x_l}(a)$ and $d_b \leftarrow \deg_{x_l}(b_l)$.
6: Let $a(\bar{x}_l) =: \Sigma_{i=0}^{d_a} a_i(\bar{x}_{l-1}) x_l^i$ be the polynomial representation of $a(\bar{x}_l)$ with respect to x_l.
7: Recursively reduce each coefficient $a_i(\bar{x}_{l-1})$ of a modJ_{l-1}:
$\tilde{a}_i(\bar{x}_{l-1}) \leftarrow$ REDUCE($a_i(\bar{x}_{l-1})$, J_{l-1}), for all $i \in \{0, \ldots, d_a\}$.
8: **while** $d_a \geq d_b$ **do**
9: $a(\bar{x}_l) \leftarrow a - \left(a_{d_a} \cdot x_l^{d_a - d_b} \cdot b_l\right)$
10: Update $d_a \leftarrow \deg_{x_l}(a)$. Update a_i's such that $a(\bar{x}_l) =: \Sigma_{i=0}^{d_a} a_i(\bar{x}_{l-1}) \cdot x_l^i$.
11: Call REDUCE($a_i(\bar{x}_{l-1})$, J_{l-1}) for all $i \in \{0, \ldots, d_a\}$: recursively reduce each coefficient $a_i(\bar{x}_{l-1})$ mod J_{l-1} (like Step 7).
12: **end while**
13: **return** $a(\bar{x}_l)$.
14: **end procedure**

Following lemma shows that reduction modulo a triangular ideal (Algorithm 2) is efficient.

Lemma 28 (Reduction)**.** *Given* $a(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]$ *and* $J_l \subseteq \mathbb{G}[\bar{x}_l]$*, to reduce* $a(\bar{x}_l)$ *mod* J_l*, Algorithm 2 takes time poly* $\left(\prod_{i=0}^{l} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_l)\right)$.

In particular, if each coefficient $a_i(\bar{x}_{l-1})$ *of* $a(\bar{x}_l)$ *(viewed as a polynomial in* x_l*) is in reduced form mod* J_{l-1}*, then reduction takes time poly*$(d_a, \log|\mathbb{G}|, \deg(J_l))$*, where* $d_a = \deg_{x_l}(a)$*.*

Proof. We prove the lemma by induction on the length $l + 1$ of the ideal J_l.

For $l = 0$, we have a standard univariate reduction which takes at most $O(\deg(a)\deg(b))$ ring operations in $\mathbb{G}$. Since addition/multiplication/division in $\mathbb{G}$ take time at most $\tilde{O}(\log|\mathbb{G}|)$ [Sho09], we get the lemma.

Assume that the lemma is true for any ideal of length less than l.

Coefficients $a_i(\bar{x}_{l-1})$ can be reduced, in time poly $\left(\prod_{i=0}^{l-1} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_{l-1})\right)$, mod J_{l-1} using induction hypothesis. We need to make $d_a + 1$ such calls; total time is bounded by poly $\left(\prod_{i=0}^{l} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_{l-1})\right)$. In the same time we can compute Step 9.

After the update at Step 9, individual-degrees $\deg_{x_i}(a)$ (for $0 \le i < l$) can become at most double the previous degree (safely assuming $2 \le \deg_{x_i}(b_i) \le \deg_{x_i}(a)$). By induction hypothesis, each call to reduce $a_i(\bar{x}_{l-1})$ mod J_{l-1} takes time poly $\left(\prod_{i=0}^{l-1} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_{l-1})\right)$. Algorithm makes at most d_a such calls and the while-loop runs at most d_a times. Hence, the algorithm takes time poly $\left(\prod_{i=0}^{l} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_l)\right)$; and we are done.

If coefficients of a are already reduced modulo J_{l-1}, then $\deg_{x_i}(a) < \deg_{x_i}(b_i)$ for all $0 \le i < l$. Hence, Algorithm 2 takes time $d_a^2 \cdot$ poly $(\log|\mathbb{G}|, \deg(J_{l-1}))$. □

Lemma 29 (Division mod triangular ideal)**.** *Given a triangular ideal $J_l \subseteq \mathbb{G}[\bar{x}_l]$ and a unit $a(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]/J_l$. We can compute a^{-1} mod J_l, in reduced form, in time poly $\left(\prod_{i=0}^{l} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_l)\right)$.*

Proof. Let $u(\bar{x}_l) \in \mathbb{G}[\bar{x}_l]/J_l$ be such that $u \cdot a \equiv 1$ mod J_l. We can write u as

$$\sum_{\substack{\bar{e} \ \ge \ \bar{0} \\ \forall\ 0 \le i \le l,\ e_i \ < \ \deg_{x_i}(b_i)}} u_{\bar{e}} \cdot \bar{x}_l^{\bar{e}} \ .$$

We want to find the unknowns $u_{\bar{e}}$ in $\mathbb{G}$, satisfying $u \cdot a \equiv 1$ mod J_l. This gives us a linear system in the unknowns; it has size $\deg(J_l)$. The linear system can be written down, using Algorithm 2, by reducing the monomial products $\bar{x}_l^{\bar{e}} \cdot \bar{x}_l^{\bar{e}'}$ that appear in the product $u \cdot a$. This takes time poly $\left(\prod_{i=0}^{l} \deg_{x_i}(a), \log|\mathbb{G}|, \deg(J_l)\right)$.

Since there exists a unique u, our linear system is efficiently solvable, by standard linear algebra, in the required time. □

Let us see two direct applications of the reduction Algorithm 2 to compute valuation and to compute reduced form of split ideals.

First, we explain how Algorithm 1 (Steps 3, 18) computes reduced f_J modulo the lift $\hat{J}$ of the newly computed split ideal J, when x is replaced by $x_{l+1} + px$ in the intermediate polynomial $f_I(\bar{x}_l, x)$.

Lemma 30 (Updating stack with reduced polynomial)**.** *Let $I \subseteq \mathbb{F}_p[\bar{x}_l]$ be a split ideal and $f_I(\bar{x}_l, x) \in R[\bar{x}_l, x]$ be reduced modulo $\hat{I}$ (the lift of I over R). Define split ideal $J \subseteq \mathbb{F}_p[\bar{x}_{l+1}]$ as $J := I + \langle h_{l+1}(\bar{x}_{l+1})\rangle$, and $\hat{J}$ be the lift of J over R.*

Then, in time poly$(\log|R|, \deg_x(f_I), \deg(J))$, we can compute a reduced polynomial f_J modulo $\hat{J}$ defined by, $f_J(\bar{x}_{l+1}, x) := f_I(\bar{x}_l, x_{l+1} + px)$ mod $\hat{J}$.

Proof. Since $f_I(\bar{x}_l, x)$ is already reduced modulo $\hat{I}$, $\deg_{x_i}(f_I) < \deg_{x_i}(h_i)$. Define $D := \deg_x(f_I)$, perform the shift $x \to x_{l+1} + px$ in f_I, and expand f_I using Taylor series,

$$f_J(\bar{x}_l, x) \ = \ f_I(\bar{x}_l, x_{l+1} + px) \ =: \ g_0(\bar{x}_{l+1}) + g_1(\bar{x}_{l+1})(px) + \ldots + g_D(\bar{x}_{l+1})(px)^D \ ,$$

where g_i could also be seen as the i-th derivative of $f_I(\bar{x}_l, x_{l+1})$ (wrt x_{l+1}) divided by $i!$. To compute f_J mod $\hat{J}$, we call REDUCE(g_i, $\hat{J}$) (for all i) to get the reduction of each term mod $\hat{J}$.

To calculate the time complexity of REDUCE(g_i, $\hat{J}$), note that coefficients of each g_i, wrt x_{l+1}, is already reduced mod $\hat{I}$. Since $J = I + \langle h_{l+1}\rangle$, using Lemma 28, time complexity of reducing each g_i by $\hat{J}$ is at most poly$(\deg_{x_{l+1}}(g_i), \log|R|, \deg(J))$ ($\deg(J) = \deg(\hat{J})$).

Since $\deg_{x_{l+1}}(g_i) \le \deg_x(f_I)$ (for $i \le D$), total time complexity is poly $(\log|R|, \deg_x(f_I), \deg(J))$. □

Next, we explain Step 20 in Algorithm 1 a bit more.

Lemma 31 (Ideal factors in reduced form)**.** *Consider the tuple* $(U := \{h_0(\bar{x}_0), \ldots, h_l(\bar{x}_l)\}, f_{\langle U\rangle}) \in S$ *and consider a non-trivial factorization* $h_i =: h_{i,1} \ldots h_{i,m}$ *for some* $h_i \in U$*. Wlog each factor* $h_{i,j}$ *is monic wrt* x_i*.*

Then, we can compute the factor-related tuples $(U_j, f_{\langle U_j\rangle})$*, for all* $j \in [m]$*, in time poly*$(\deg(\langle U\rangle), \log|R|, \deg_x(f_{\langle U\rangle}))$ *(*$f_{\langle U_j\rangle}$ *will be in reduced form mod* $\langle U_j\rangle$*).*

Proof. First, we successively reduce h_{i+t} $(1 \leq t \leq l-i)$ modulo triangular ideal $I_{i+t,j} := \langle h_0, \ldots, h_{i-1}, h_{i,j}, h_{i+1}, \ldots, h_{i+t}\rangle$. Time complexity of each of these steps is bounded by poly$(\deg(\langle U\rangle), \log|R|)$ (Lemma 28). This ensures that the degree of h_{i+t} in a variable x_s $(s < i+t)$ is less than the individual-degree of the s-th generator of ideal $\langle U_j\rangle$.

Then, $f_{\langle U_j\rangle}$ can be calculated by reducing each $\deg_x(f_{\langle U\rangle})+1$ coefficients of $f_{\langle U_j\rangle}$ (wrt x) by the lifted triangular ideal $\hat{I}_{l,j} = \hat{U}_j$. By Lemma 28, this takes time poly$(\prod_{i=0}^{l}\deg_{x_i}(f_{\langle U\rangle}), \deg_x(f_{\langle U\rangle}), \log|R|, \deg(\langle U\rangle))$. Since coefficients (wrt x) of $f_{\langle U\rangle}$ were already reduced modulo $\langle U\rangle$, $\prod_{i=0}^{l}\deg_{x_i}(f_{\langle U\rangle}) \leq \deg(\langle U\rangle)$.

So, the computation time is bounded by poly $\big(\deg(\langle U\rangle), \log|R|, \deg_x(f_{\langle U\rangle})\big)$. □

D Computation modulo a triangular ideal— Zerodivisor test & GCD

Test-Zero-Div$(a(\bar{x}_l), I_l)$, for a triangular ideal $I_l =: \langle h_0, \ldots, h_l\rangle$, either reports that $a(\bar{x}_l)$ is not a zerodivisor modulo I_l, or *returns a non-trivial factorization* of a generator $h_i =: h_{i,1} \cdots h_{i,m}$ (into monic, wrt x_i, factors mod prior ideal). In this section we assume $\mathbb{F}$ to be a finite field.

Idea: In the quotient ring $\mathbb{F}[\bar{x}_l]/\langle I_l\rangle$, a monic (wrt x_i) polynomial $a(\bar{x}_i)$ is a zerodivisor iff it contains a factor of $h_i(\bar{x}_i)$— generator of triangular ideal I_l with variables $\{x_0, \ldots, x_i\}$. So, firstly the algorithm checks if the given polynomial $a(\bar{x}_l)$ is monic (recursively, from variables x_{l-1} to x_0). If it fails, it factors some generator h_i for $i < l$. After making $a(\bar{x}_l)$ monic, we take gcd of a with h_l— if it finds non-trivial gcd it factors h_l, else $a(\bar{x}_l)$ is not a zerodivisor.

Algorithm 3 Zerodivisor test of $a(\bar{x}_l)$ modulo I_l

1: **procedure** Test-Zero-Div$(a(\bar{x}_l), I_l)$
2: **if** $l = 0$ **then**
3: [**Take univariate GCD**] $gcd \leftarrow \gcd(a(x_0), h_0(x_0))$.
4: **if** gcd is non-trivial **then**
5: Factorize $h_0(x_0) =: gcd \cdot \frac{h_0}{gcd}$; **return** $(True, gcd \cdot \frac{h_0}{gcd})$.
6: **else**
7: **return** $(False)$.
8: **end if**
9: **end if**
10: Let the leading coefficient of $a(\bar{x}_l)$ wrt x_l be $\tilde{a}(\bar{x}_{l-1})$.
11: Call Test-Zero-Div$(\tilde{a}(\bar{x}_{l-1}), I_{l-1})$.
12: **if** The test returned $True$ **then**
13: **return** the result of the test including the factorization of a generator $h_i(\bar{x}_i)$.

14: **end if**

[Now, we will take gcd of a and h_l using iterated division method (Euclid's method).]

15: Define $b(\bar{x}_l) \leftarrow h_l(\bar{x}_l)$.

16: **while** $b(\bar{x}_l) \neq 0$ **do**

17: Let $\tilde{b}(\bar{x}_{l-1})$ be the leading coefficient of $b(\bar{x}_l)$ wrt x_l.

18: **if** TEST-ZERO-DIV($\tilde{b}(\bar{x}_{l-1})$, I_{l-1}) $= True$ **then**

19: **return** result of TEST-ZERO-DIV($\tilde{b}(\bar{x}_{l-1})$,I_{l-1}), factorization of a generator $h_i(\bar{x}_i)$.

20: **end if**

21: Let $c(\bar{x}_l) \leftarrow$ REDUCE($a(\bar{x}_l)$, $I_{l-1} + \langle b(\bar{x}_l)/\tilde{b}\rangle$) (same as taking remainder of $a(\bar{x}_l)$ when divided by the monic polynomial $b(\bar{x}_l)/\tilde{b}$ modulo I_{l-1}).

22: $a(\bar{x}_l) \leftarrow b(\bar{x}_l)/\tilde{b}$, $b(\bar{x}_l) \leftarrow c(\bar{x}_l)$. [Invariant: $\deg_{x_l}(b)$ has fallen.]

23: **end while**

[Gcd of original $a(\bar{x}_l)$ and $h_l(\bar{x}_l)$ mod I_l is stored in $a(\bar{x}_l)$.]

24: **if** gcd $a(\bar{x}_l)$ is non-trivial **then**

25: **return** ($True$, a non-trivial factorization of $h_l(\bar{x}_l)$).

26: **else**

27: **return** ($False$). [$a(\bar{x}_l)$ is not a zerodivisor.]

28: **end if**

29: **end procedure**

Lemma 32 (Efficiency of testing zerodivisors)**.** *Assuming, coefficients of $a(\bar{x}_l)$ wrt x_l are in reduced form modulo I_{l-1}, Algorithm 4 takes time poly*$(\deg_{x_l}(a), \log|\mathbb{F}|, \deg(I_l))$.

Proof. We apply induction on the length $l+1$ of ideal I_l.

For $l = 0$, it runs univariate gcd and takes time $\text{poly}(\deg(a), \deg(h_0), \log|\mathbb{F}|)$ [Sho09].

Assume lemma statement holds true for ideals of length l.

By induction, checking $\tilde{a}(\bar{x}_{l-1})$ is a zerodivisor mod I_{l-1}, takes $\text{poly}(\deg_{x_{l-1}}(\tilde{a}), \log|\mathbb{F}|, \deg(I_{l-1}))$ time.

To compute gcd of a and h_l, Euclidean gcd algorithm will run at most $\deg_{x_l}(a) + \deg_{x_l}(h_l)$ while-loops. From induction hypothesis, and Lemmas 28-29, each loop takes at most $\text{poly}(\deg_{x_l}(a), \log|\mathbb{F}|, \deg(I_l))$ time. So, we are done. □

GCD($a(\bar{x}_l, x)$, $b(\bar{x}_l, x)$, I_l) computes gcd of two polynomials $a(\bar{x}_l, x)$ and $b(\bar{x}_l, x)$ modulo a triangular ideal $I_l = \langle h_0(x_0), \ldots, h_l(\bar{x}_l)\rangle$ resp. $False$. It computes the *monic gcd* resp. returns a *non-trivial factorization* of some h_i.

Algorithm 4 GCD computation modulo I_l

1: **procedure** GCD($a(\bar{x}_l, x)$, $b(\bar{x}_l, x)$, I_l)

2: Let $\tilde{b}(\bar{x}_l)$ be the leading coefficient of b with respect to x.

3: **if** TEST-ZERO-DIV($\tilde{b}(\bar{x}_l)$, I_l) $= True$ **then**

4: **return** $False$, TEST-ZERO-DIV($\tilde{b}(\bar{x}_l)$, I_l) factors some generator $h_i(\bar{x}_i)$.

5: **end if**

6: Let $c(\bar{x}_l, x) \leftarrow$REDUCE($a$, $I_l + \langle b/\tilde{b}\rangle$).

7: **if** $c = 0$ **then**

8: **return** $b/\tilde{b}$.

9: **else**

10: **return** GCD($b(\bar{x}_l, x)$, $c(\bar{x}_l, x)$, I_l).
11: **end if**
12: **end procedure**

Lemma 33 (Multivariate GCD)**.** *Algorithm 4 either factors a generator h_i (& outputs False), or computes a monic polynomial $g(\bar{x}_l, x) \in \mathbb{F}[\bar{x}_l, x]$, such that, g divides a, b modulo I_l. Moreover, $g = ua + vb \bmod I_l$, for some $u(\bar{x}_l, x), v(\bar{x}_l, x) \in \mathbb{F}[\bar{x}_l, x]$.*

If a and b are in reduced form mod I_l, then it takes time poly$(\deg_x(a), \deg_x(b), \log|\mathbb{F}|, \deg(I_l))$.

Proof. Algorithm 4 is just an implementation of multivariate Euclidean gcd algorithm over the coefficient ring $\mathbb{F}_p[\bar{x}_l]/I_l =: R'$. If the algorithm outputs $g(\bar{x}_l, x) \in R'[x_l]$ then, by standard Euclidean gcd arguments (using recursion), there exists $u(\bar{x}_l, x), v(\bar{x}_l, x) \in R'[x]$, such that, $ua + vb = g$, and g divides both a and b modulo I_l.

The algorithm works fine if in each step it was able to work with a monic divisor. Otherwise, it gets stuck at a 'division' step, implying that the divisor's leading-coefficient is a zerodivisor, factoring some generator of I_l.

For time complexity, each recursive step makes one call each to Test-Zero-Div, Reduce, and division procedures. They take time poly$(\deg_x(a), \deg_x(b), \log|\mathbb{F}|, \deg(I_l))$ ($\because$ coefficients of a and b are in reduced form mod I_l, and use Lemmas 28, 29 & 32). Since number of recursive steps are bounded by $\deg_x(a) + \deg_x(b)$, total time is bounded by poly$(\deg_x(a), \deg_x(b), \log|\mathbb{F}|, \deg(I_l))$. □