Checking Polynomial Identities over any Field: Towards a Derandomization?

Daniel Lewin*
Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge, MA02139.
danl@theory.lcs.mit.edu.

Salil Vadhan†
Department of Mathematics,
Massachusetts Institute of Technology,
Cambridge, MA02139.
salil@math.mit.edu.

November 13, 1997

Abstract

We present a Monte Carlo algorithm for testing multivariate polynomial identities over any field using less random bits than other methods. To test if a polynomial $P(x_1, \ldots, x_n)$ is zero, our method uses $\sum_{i=1}^{n} \lceil \log(d_i + 1) \rceil$ random bits , where d_i is the degree of x_i in P, to obtain *any* inverse polynomial error in polynomial time. The algorithm applies to polynomials given as a black box or in some implicit representation such as a straight line program. Our method works by evaluating P at truncated formal power series representing square roots of irreducible polynomials over the field. This approach is similar to [CK97], but with the advantage that the techniques are purely algebraic and apply to any field.

We view uncovering this algebraic structure as a step towards the derandomization of polynomial identity testing, a long standing open question.

*Supported by the love of his wife and kids.
†Supported by an NDSEG/DOD Graduate Fellowship.

1 Introduction

Checking multivariate polynomial identities of the form $P_1(x_1, \ldots, x_n) = P_2(x_1, \ldots, x_n)$ is a problem central to both algorithm design and complexity theory. Algorithms such as the RNC algorithm for perfect matching [Lov79, MVV87, CRS95], the BPP algorithm for testing equivalence of read-once branching programs [BCW80], and one of the randomized algorithms for testing multiset equality [BK95] rely on efficiently checking if a multivariate polynomial is identically zero. Results in complexity theory such as IP = PSPACE [LFKN90, Sha90], MIP = NEXPTIME [BFL90], and NP = PCP($\log n$, 1) [AS92, ALM+92] all fundamentally rely on viewing a boolean assignment not as a group of bits, but as the values of a multivariate polynomial. Testing if such a polynomial is identically zero is a procedure used frequently in this context. In addition, many results in learning theory, and sparse multivariate polynomial interpolation also rely on checking polynomial identities [Zip79, GKS90, CDGK91, RB91].

Clearly, the problem is easy if the input polynomials are given as lists of coefficients (known as standard reduced form). However, in many cases the polynomials are given in some implicit representation such as a symbolic determinant or as a product of multiple polynomials. Reducing a polynomial in such a succinct representation to its standard form can take exponential time in the length of the description since there could be an exponential number of non-zero coefficients that need to be determined. A property of many succinct representations is that despite the fact that the reduced standard form of the polynomial may have exponential size, it is possible to evaluate the polynomial at a given point in only polynomial time. For example, the determinant can be evaluated in polynomial time, as can a polynomial sized product of polynomials.

Many randomized methods for checking polynomial identities have been discovered based on the assumption that the polynomials can be evaluated efficiently. The basic scheme is to use randomization to select a number of sample points on which the identity is checked by evaluation. The test accepts if the identity is found to hold at all the sample points and rejects otherwise. Schwartz and Zippel discovered in [Sch80] and [Zip79] that the probability that a non-zero multivariate polynomial evaluates to zero is small as long as the point is selected at random from a large enough domain. In a recent development, Chen and Kao [CK97] showed how to check if a polynomial with integer coefficients is zero using fewer random bits than the Schwartz-Zippel method. Their method is to evaluate the polynomial at approximations of easily computable irrational points. An innovative feature of Chen and Kao's algorithm is that the error probability of the test can be decreased by doing more computations instead of increasing the number of random bits used. The main drawback of Chen and Kao's algorithm is that it only applies to polynomials with *integer* coefficients.

In this paper we extend Chen and Kao's work by showing how to achieve the same result in *any* field. Our result is obtained by uncovering the essential ingredients of Chen and Kao's algorithm and abstracting them. We obtain a purely algebraic formulation of the algorithm while Chen and Kao's description relies on the structure of the real numbers. We view uncovering this algebraic structure as a step towards the derandomization of polynomial identity checking.

Using the Schwartz-Zippel lemma and a simple counting argument, one can show that there exists a set S of poly(s, d) points, so that any nonzero multivariate polynomial of 'description size' at most s and degree at most d evaluates to non-zero on at least one of the points of S. Finding such a set of points *deterministically* would be a major breakthrough, as it would imply the derandomization of all polynomial identity testing, a long standing open problem. Even for the case in which P is restricted to symbolic determinants with entries that are linear forms in the input variables, it is not known how to construct such a set explicitly.

We view our work (as well as that of [CK97]) as restricting the domain in which one has to search for a set of "good points." Our purely algebraic approach, in contrast to that of [CK97], results in a highly structured domain, whose algebraic properties might give insight into the search for good evaluation points.

1.1 Previous Algorithms

Let F be a field. For most of the paper, we assume that a multivariate polynomial $P(x_1, \ldots, x_n)$ with coefficients in F is described by an efficient procedure for evaluating P given values for $x_1, \ldots, x_n$. Such a procedure can, for example, be described by a *straight line program* doing computations in F. For example, P could be a symbolic determinant over F, and the procedure would be any efficient method for computing the determinant. In Appendix A we discuss straight-line programs as well as the "Black Box" model, in which P is represented by a black box that given values for $x_1, \ldots, x_n$ evaluates P at that point.

We concentrate on algorithms for checking if the polynomial $P(x_1, \ldots, x_n)$ is zero since any polynomial identity can be transformed into this form.

1.1.1 Schwartz-Zippel

The first randomized test was discovered both by Schwartz and Zippel. The method is based on the following famous lemma.

Lemma 1.1 ([Sch80, Zip79]) *Let d be the degree of $P(x_1, \ldots, x_n)$. Let S be a set of size at least Cd. If P is not identically zero, then $P(s_1, \ldots, s_n) = 0$ with probability at most $\frac{1}{C}$, where $s_1, \ldots, s_n$ are chosen uniformly and at random from S.*

This lemma immediately implies the following test:

1. Choose a random point $(s_1, \ldots, s_n)$ from S^n, where $S \subseteq F$, and $|S| = 2d$.
2. Evaluate $P(s_1, \ldots, s_n)$ using the procedure supplied for P.
3. Output `'nonzero'` if $P(s_1, \ldots, s_n) \neq 0$, else output `'probably zero'`.

One technicality is that if the field F has fewer than $2d$ elements in it, then there is no set S large enough to be used in the algorithm. In this case, S can be selected from an extension field of F and P is evaluated over the extension field. In Section A, we discuss how the given procedure for evaluating P can be modified to evaluate P over an extension field.

Clearly if $P = 0$, the test always outputs `'probably zero'` which is the correct answer. On the other hand, Lemma 1.1 implies that if $P \neq 0$, then the test is wrong with probability no more than $\frac{1}{2}$. That is, the error probability is at most $\frac{1}{2}$. The algorithm clearly uses $n\lceil \log 2d \rceil$ random bits.

As discussed in [CK97], there are three basic methods to reduce the error probability of the Schwartz-Zippel algorithm to $1/t$ for an arbitrary t. The first is to perform $\lceil \log t \rceil$ independent repetitions of the above test, using $\lceil \log t \rceil n \lceil \log 2d \rceil$ random bits. The second is to enlarge the size of S to be td (possibly moving to an extension field of F) thus using $n\lceil \log td \rceil$ random bits. The third, which works for $t \leq 2^{n\lceil \log 2d \rceil}$ is to perform t pairwise independent repetitions of the algorithm, thus using $2n\lceil \log 2d \rceil$ random bits.

1.1.2 Chen-Kao

Recently, Chen and Kao [CK97] discovered a new algorithm for testing if a multivariate polynomial is identically zero. Their algorithm uses fewer random bits than the algorithm of Schwartz-Zippel in order to obtain a given error probability. Chen and Kao's algorithm only works for polynomials with *integer* coefficients.

Chen and Kao's basic strategy is to evaluate the polynomial $P(x_1, \ldots, x_n)$ at a set of irrational points $\pi_1, \ldots \pi_n \in \mathbb{R}$. In their algorithm, each π_i is a sum of a small number of square roots of primes: $\pi_i = \sum_{i,j} \sqrt{p_{ij}}$. They show that $P(\pi_1, \ldots, \pi_n) = 0$ if and only if P is identically zero. That is, if you can evaluate the polynomial P at this single point, then you can check if P is identically zero!

Unfortunately, this does not immediately imply a testing algorithm since P needs to be evaluated at infinite precision irrational numbers. To get around this problem, Chen and Kao approximate each $\sqrt{p_{ij}}$ by r_{ij} which is obtained by truncating the binary expansion of $\sqrt{p_{ij}}$ at the ℓ'th position. They then show that if P is evaluated at the points $\overline{\pi}_i = \sum_{i,j} \sigma_{ij} r_{ij}$ where σ_{ij} is *randomly chosen* to be $+1$ or -1, then the error probability drops proportionately to $1/\ell$. This implies the surprising result that any inverse polynomial error can be achieved in polynomial time while using the same number of random bits!

For reference, we roughly describe the Chen-Kao algorithm below:

Let d_i be the degree of x_i in P.

1. **Find Primes:** Find the first $\sum_i \log(d_i + 1)$ primes p_{ij}, $1 \leq i \leq n$, $1 \leq j \leq \log(d_i + 1)$.
2. **Approximate Square Roots:** Compute the r_{ij}'s by computing the first ℓ bits of $\sqrt{p_{ij}}$.
3. **Add Randomization:** Set $\overline{\pi}_i = \sum_{i,j} \sigma_{ij} r_{ij}$ where σ_{ij} is randomly chosen to be $+1$ or -1.
4. **Evaluate Polynomial:** Output `'nonzero'` if $P(\overline{\pi}_1, \ldots, \overline{\pi}_n) \neq 0$, else output `'probably zero'`.

From this description, we see that the Chen-Kao algorithm uses $\sum_i \log_2(d_i + 1)$ random bits to achieve any inverse polynomial error probability in polynomial time. This can be substantially lower than the number of random bits used by Schwartz-Zippel, which is at least $n \log_2(2d)$ to achieve an error probability of $1/2$. In the simple case that P is a multilinear polynomial of degree n, Chen-Kao use n random bits compared to $n \log n$ for Schwartz-Zippel.

1.1.3 Our Contribution

At first glance, it seems that the techniques of Chen and Kao cannot be extended to finite fields, since there are no clear notions of primes or approximations in finite fields. This seems to imply that testing polynomial identities over the integers is somehow easier than over an arbitrary field.

In this paper we show that this is not the case. We obtain results comparable to those of Chen and Kao that hold for polynomials with coefficients from any field F. More specifically, we show that over any field F it is possible to test if a multivariate polynomial $P(x_1, \ldots, x_n)$ is zero with *any* inverse polynomial error probability in polynomial time, using only $\sum \log_2(d_i + 1)$ random bits (d_i is the degree of x_i in P).

The first obstacle in extending Chen and Kao's approach is the lack of "primes" in arbitrary fields (or even in finite fields). We overcome this by extending our view from the field F to the ring of polynomials $F[x]$. Now, it seems natural that irreducible polynomials over F take the place of the primes in Chen and Kao's algorithm. But what is a square root of an irreducible polynomial? Clearly, irreducible polynomials do not have square roots that are polynomials, but it turns out that they may have roots which are *infinite power series*![1]

For example, consider the polynomial $x + 1$ over the field with three elements. The square root of this polynomial as an infinite power series is:

$$1 + 2x + x^2 + x^3 + 2x^4 + \ldots$$

This notion of a root implies a natural extension of the notion of approximation. Namely, approximations are obtained by truncating infinite power series at some power x^ℓ (which can be viewed as taking the series *modulo* x^ℓ). For example, the approximation of the square root of $x + 1$ modulo x^2 in the field of three elements is the *polynomial* $1 + 2x$.

[1] This is assuming that the field is not of characteristic 2. This case is treated in Section 5.

Thus, the intuition behind our algorithm can be summed up in the following table:

Primes	$\rightarrow$	Irreducible Polynomials in $F[x]$
Square Roots	$\rightarrow$	Infinite Power Series over F
Approximation	$\rightarrow$	Square Roots mod x^ℓ

Using this analogy, a rough description of our algorithm reads much the same as Chen and Kao's algorithm:

1. **Find Irreducible Polynomials:** Find $\sum_i \log(d_i+1)$ distinct irreducible polynomials p_{ij} ($1 \leq i \leq n$, $1 \leq j \leq \log(d_i+1)$ that have square roots as infinite power series.
2. **Approximate Square Roots:** Compute approximations r_{ij} to the square roots $\sqrt{p_{ij}}$ modulo x^ℓ. Note that r_{ij} is a polynomial.
3. **Add Randomization:** Set $\overline{\pi}_i = \sum_{ij} \sigma_{ij} r_{ij}$ where σ_{ij} is randomly chosen to be $+1$ or -1. Note that the $\overline{\pi}_i$ are polynomials!
4. **Evaluate Polynomial:** Output `'nonzero'` if $P(\overline{\pi}_1, \ldots, \overline{\pi}_n) \neq 0 \pmod{x^\ell}$. Note that we evaluate P after a *univariate* polynomial has been substituted in place of each of its variables.

We show that the error probability of this test can be reduced, in polynomial time, to *any* inverse polynomial quantity by using approximations modulo larger powers of x.

1.2 Layout of the Paper

Section 2 describes some standard algebraic tools which our algorithm uses. Section 3 gives a more detailed description of our algorithm, along with an example. In Section 4, we prove the correctness of our algorithm; this section contains most of the technical contributions of this paper. We feel that the analysis of the algorithm makes use of techniques that may be useful in other applications involving multivariate polynomials.

2 Algebraic Tools

In this section we describe the basic algebraic procedures that are used in the algorithm. We describe procedures for:

1. Finding irreducible polynomials that have square roots as power series.
2. Finding approximations to the square roots of irreducible polynomials.

Our goal in this section is to show efficient algorithms for each of the above tasks. However, in the interest of clarity, we do not always describe the most efficient algorithms that are known.

In this section we assume that the field we are working over is not of characteristic 2. The case of characteristic 2 is dealt with in Section 5.

2.1 Definitions

Let F be a field of characteristic $\neq 2$. We denote by $F[x]$ the ring of polynomials over the field F, and by $F(x)$ the field of fractions of $F[x]$; in other words, $F(x)$ is the field of rational functions over F. The ring of formal power series over F is denoted $F[[x]]$. We denote by $F[\alpha]$ the field extension of F obtained by adjoining to F an algebraic element α.

2.2 Finding Irreducible Polynomials and Approximating Square Roots

Not every irreducible polynomial has a square root as an formal power series. For example, over the rationals, the polynomial $x-3$ is irreducible, but does *not* have a square root as an formal power series since the constant term of the series has to be $\sqrt{3}$, which is irrational. This example shows that for an irreducible polynomial to have a square root, its constant term must be a quadratic residue. Surprisingly, in fields of characteristic $\neq 2$, this condition is also sufficient! This is a special case of a very useful construction called *Hensel Lifting*.

Hensel Lifting is described in two parts. First we state Hensel's Lemma which characterizes when a polynomial equation with coefficients in $F[x]$ has a root in $F[[x]]$. For example, finding a square root of a polynomial $f(x) \in F[x]$ can be viewed as finding a root in $F[[x]]$ of $Z^2 - f = 0$. In Appendix D, we describe a standard technique for finding approximations to these roots, given that they exist.

Lemma 2.1 (Hensel's Lemma [Eis95, Cor. 7.4]) *Let $S(Z)$ be a polynomial with coefficients in $F[x]$. S can be viewed as a bivariate polynomial $S(Z,x)$ over F. If there is a $g \in F$ such that:*

1. $S(g,0) = 0$.
2. $S_Z(g,0) \neq 0$ *where* $S_Z(Z,x) = \frac{\partial S(Z,x)}{\partial Z}$.

Then, there exists a $\tilde{g}(x) \in F[[x]]$ such that $S(\tilde{g}(x),x) = 0$.

Say we have an irreducible polynomial $f(x) \in F[x]$. We can use Lemma 2.1 to find the conditions under which $f(x)$ has a square root in $F[[x]]$. Let $S(Z,x)$ be the polynomial $S(Z,x) = Z^2 - f(x)$. The two conditions of Lemma 2.1 are:

1. $S(g,0) = g^2 - f_0 = 0$ where f_0 is the constant term of f. So, g is a square root of f_0 in F.
2. $S_Z(g,0) = 2g \neq 0$. This is true as long as $f_0 \neq 0$, and F is not of characteristic 2.

So, from Lemma 2.1 and our previous discussion it follows that $f(x)$ has a square root as an formal power series if and only if f_0 is a quadratic residue over F.

A proof of an even more general "Hensel's lemma", which implies Lemma 2.1, can be found in [Eis95, Ch. 7].

2.2.1 Finding Irreducible Polynomials With Square Roots

Lemma 2.1 tells us that any irreducible polynomial with a constant term that is a quadratic residue in F, has a square root in $F[[x]]$ (assuming that F is not of characteristic 2). As a subroutine of our algorithm we need to be able to find k such polynomials in $F[x]$. Clearly, if we can find k distinct, monic, irreducible polynomials in $F[x]$, then by multiplying each by its constant term we obtain a set of k irreducible polynomials that have square roots in $F[[x]]$!

Luckily, finding k monic, irreducible polynomials in F is not hard. In fact, if we don't care about being as efficient as possible we can just hunt for them by brute force. That is, go through the monic elements of $F[x]$ one by one in order of degree, and check if any of the irreducible polynomials found so far divides them. If none do, we have another irreducible polynomial, otherwise we move on to the next element of $F[x]$. The following lemma says that if we want to find k polynomials this way, we don't have to search very far.

Lemma 2.2 [2] *Let F be a finite field, and $F[x]$ the ring of polynomials over F. Then, the number of irreducible polynomials of degree at most n in $F[x]$ is at least $(|F|^n - 1)/n$.*

[2] Actually, in analogy to the famous Prime Number Theorem over $\mathbb{Z}$, it is known that the number of irreducible polynomials of degree n over F is asymptotic to $|F|^n/n$.

The proof of Lemma 2.2 is in Appendix E.

If $k \leq |F|$, we only need to use degree 1 polynomials: $x-e_1, x-e_2, \ldots, x-e_k$ where $e_i \in F$. However, if $k > |F|$ then Lemma 2.2 says that we do not have to go over more than polynomial in k elements of $F[x]$ until we find k monic, irreducible polynomials.

We have seen how to find a set of irreducible polynomials that have square roots as power series, but how can we find approximations of the square roots efficiently? Luckily, there is a well known method for finding square roots modulo x^ℓ using $\text{poly}(\ell)$ algebraic operations in F. This is described in Appendix D.

Hensel lifting has been used for other algorithmic purposes, such as factoring sparse multivariate polynomials [Zip79, Zip81, Kal82, vzGK85].

3 The Algorithm

In this section we give a formal description of our algorithm for testing if a multivariate polynomial is zero. The algorithm is described and then a simple example of how the algorithm runs is presented. The proof of correctness of the algorithm is in Section 4.

Inputs to the algorithm:

1. A multivariate polynomial $P(x_1, \ldots, x_n) \in F[x_1, \ldots, x_n]$ described by a "straight-line program". The algorithm checks if P is zero.
2. Upper bounds d_i, $1 \leq i \leq n$ on the maximum degree of x_i in the polynomial P, and an upper bound d on the total degree of P.
3. The desired probability of error, ϵ, $0 < \epsilon \leq 1$.

The notion of a straight line program is defined and discussed in Appendix A.1 . In Appendix A.2, we discuss our algorithm in the "Black Box" model, in which the polynomial P is given as a black box that can evaluate P at points in F^n.

In most applications, the structure of P can be used to obtain the degree bounds. For example, if P is a symbolic determinant, then the degree of any variable is not more that the number if times it appears in the matrix (with multiplicity).

The algorithm: On input: P, d, d_i, $1 \leq i \leq n$, and ϵ:

Find Irreducible Polynomials: Find $\sum_i \lceil \log(d_i+1) \rceil$ irreducible polynomials p_{ij} ($1 \leq i \leq n$, $1 \leq j \leq \lceil \log(d_i+1) \rceil$) that have square roots as infinite power series. Do this by computing by brute force the first $\sum_i \lceil \log(d_i+1) \rceil$ monic, irreducible polynomials and multiplying them by their constant term.

Approximate Square Roots: Set $\ell = \frac{1}{\epsilon}\left(\frac{d \max(\deg(p_{ij}))}{2}\right)$. Compute approximations r_{ij} to the square roots of the p_{ij}'s modulo x^ℓ, using the Hensel Lifting algorithm described in Section 2. That is, compute $r_{ij} = \sqrt{p_{ij}}$ modulo x^l.

Add Randomization: Set $\overline{\pi}_i = \sum_{ij} \sigma_{ij} r_{ij}$ where σ_{ij} is randomly chosen to be $+1$ or -1.

Evaluate Polynomial: Output `'nonzero'` if $P(\overline{\pi}_1, \ldots, \overline{\pi}_n) \neq 0 \pmod{x^\ell}$, else output `'probably zero'`. Note that we evaluate P after a univariate polynomial has been substituted in place of each of its variables. Appendix A.1 discusses how to modify the straight-line program for P to do this evaluation efficiently.

It is easy to show that the running time of the algorithm is polynomial in d, n, $1/\epsilon$, and the description size of P. This is discussed in more detail in Appendix B.

The algorithm uses $\sum_i \lceil \log(d_i + 1) \rceil$ random bits and in Section 4, we show that the error probability of the algorithm is no more than ϵ. Note that as in Chen and Kao's algorithm, we can decrease the error probability without using a single additional random bit!

3.1 An Example

The following example is meant to clarify how the algorithm works.

Suppose that you are walking down the street, minding your own business, when suddenly you hear a wispy, "Psssst! Wanna buy a multivariate polynomial identity?" You turn and see a shady character emerge from a dark alleyway. He opens a side of his jacket and hanging inside, next to the cheap watches, is a straight line program for the polynomial:

$$x_1 x_2 + 2x_1 + 2x_2 + 1$$

The man claims that this polynomial is zero over the field of three elements, and offers to sell it to you for an extravagant price. Of course, you want to take the identity for a test drive before you make the purchase. You reach into your left pocket, and are dismayed to find that you forgot all but two of your random bits at home! Luckily, you have just finished implementing the algorithm described in this paper on your Palm Pilot. After punching in the polynomial and feeding in the random bits, this is what happens...

The polynomial is multilinear, so only two irreducible polynomials are needed. Searching by brute force gives: $x + 1$ and $x^2 + 1$. The power series roots of these polynomials are:

$$\sqrt{x+1} = 1 + 2x + x^2 + x^3 + 2x^4 + \cdots$$
$$\sqrt{x^2+1} = 1 + 2x^2 + x^4 + x^6 + 2x^8 + \cdots$$

Now, set $\pi_1 = \sigma_1\sqrt{x+1} \pmod{x^\ell}$, and $\pi_2 = \sigma_2\sqrt{x^2+1} \pmod{x^\ell}$ for $\sigma_i = \pm 1$. The algorithm outputs `'nonzero'` if $\pi_1\pi_2 + 2\pi_1 + 2\pi_2 + 1 \pmod{x^\ell} \neq 0$. To see how the algorithm works, we try this for $\ell = 1, 2, 3, 4$, and get the following table:

σ_1	σ_2	$\pmod{x}$	$\pmod{x^2}$	$\pmod{x^3}$	$\pmod{x^4}$
1	1	$= 0$	$= 0$	$= 0$	$\neq 0$
1	-1	$= 0$	$\neq 0$	$\neq 0$	$\neq 0$
-1	1	$= 0$	$= 0$	$\neq 0$	$\neq 0$
-1	-1	$= 0$	$\neq 0$	$\neq 0$	$\neq 0$

Note that as we use better approximations of the square roots and compute modulo larger powers of x, the probability of error (taken over the choice of σ_1, and σ_2), goes down. The number of random bits stays the same!

4 Analysis

In this section, we prove that the algorithm presented in the previous section works. That is, we show that if the input polynomial is the zero polynomial, then the algorithm always outputs `'probably zero'`. On

the other hand, if the polynomial is not identically zero, then we show that the algorithm makes a mistake with probability less than ϵ. Formally, we have

Theorem 4.1

1. *If $P(x_1, \ldots, x_n)$ is the zero polynomial, then the algorithm always outputs* `'probably zero'`.
2. *If $P(x_1, \ldots, x_n)$ is* not *zero, the probability that the algorithm outputs* `'probably zero'` *is no more than ϵ.*

The proof of Theorem 4.1 contains most of the technical contributions of this paper.

Proof: If P is the zero polynomial in $F[x_1, \ldots, x_n]$, then substituting the π_i's in place of the x_i's produces the zero polynomial in $F[x]$, which is zero modulo x^ℓ. Therefore, no matter what ℓ and the σ_i are, the algorithm outputs `'probably zero'`.

The first basic concept is that we extend our view from the field F to $F[x]$, and then to the field of fractions $F(x)$ (elements of $F(x)$ can be viewed as rational functions in x). Now, the polynomials $Z^2 - p_{ij}$ are irreducible in the ring $F(x)[Z]$, because the p_{ij} are irreducible in $F[x]$. Hence, we can look at the field extension of $F(x)$ obtained by adjoining to $F(x)$, all the elements $\sqrt{p_{ij}}$ which are the roots of the polynomials $Z^2 - p_{ij}$. This extension is denoted $K = F(x)[\sqrt{p_{ij}}]$.

The proof of Theorem 4.1 relies on the following lemma. Roughly, the lemma states that if we evaluate the polynomial P over *infinite* power series, instead of truncated ones, then the algorithm always correctly identifies polynomials that are non-zero.

Throughout the proof, we write e_i for $\lceil \log(d_i + 1) \rceil$ and denote by M the value $\sum_{i=1}^{n} e_i$.

Lemma 4.2 *Let σ_{ij} be $+1$ or -1, for $1 \leq i \leq n$, $1 \leq j \leq e_i$. For $1 \leq i \leq n$, let $q_i = \sum_{j=1}^{e_i} \sigma_{ij} \sqrt{p_{ij}}$. Then, if $P(x_1, \ldots, x_n)$ is a non zero polynomial in $F[x_1, \ldots, x_n]$, then $P(q_1, \ldots, q_n) \neq 0$ in K.*

For example, the polynomial $x_1x_2+2x_1+2x_2+1$ over the field with three elements (reusing the example of Section 3.1) is not zero in $F[x_1, x_2]$. Lemma 4.2 states that if we evaluate $\sqrt{x+1}\sqrt{x^2+1}+2\sqrt{x+1}+2\sqrt{x^2+1}+1$ in the field extension $F(x)[\sqrt{x+1}, \sqrt{x^2+1}]$, then we get a non-zero value.

Proof:

We prove Lemma 4.2 by induction on n (the number of variables in the polynomial). For $n = 0$, the result is trivial, so we consider $n > 1$. Given $P(x_1, \ldots, x_n)$, we rewrite the polynomial as:

$$P(x_1, \ldots, x_n) = \sum_{i=1}^{d_n} x_n^i P_i(x_1, \ldots, x_{n-1})$$

Now, since we assume that P is not zero, at least one of the $P_i(x_1, \ldots, x_{n-1})$ must be non-zero. Hence, by the induction hypothesis we have that:

$$P(q_1, \ldots, q_{n-1}, x_n) = \sum_{i=1}^{d_n} x_n^i P_i(q_1, \ldots, q_{n-1})$$

is a non-zero univariate polynomial in x_n of degree no more than d_n (with coefficients in $F(x)[q_1, \ldots, q_{n-1}]$). The following claim demonstrates that q_n cannot be a root of this polynomial:

Claim 4.3 *q_n is of degree $\geq d_n + 1$ over $F(x)[q_1, \ldots, q_{n-1}]$.*

The proof of Claim 4.3 is in Appendix E.

For each possible selection of the signs σ_{ij}, we call $P(q_1, q_2, \ldots, q_n)$ a *conjugate* of P. Using this terminology, choosing random σij's can be viewed as choosing a *random conjugate* from the 2^M possible conjugates.

Lemma 4.2 says that if we could compute efficiently with infinite power series, then, no matter which conjugate we choose (by choosing the σ_{ij}), $P(q_1, \ldots, q_n)$ is non-zero as long as P is non-zero. However, since we cannot compute using *infinite* power series we truncate the q_i by doing all operations modulo x^ℓ. Thus, the algorithm can viewed as evaluating a *random conjugate* modulo x^ℓ.

The statement of Theorem 4.1 can therefore be restated as: If P is a non-zero polynomial, then not more than an ϵ fraction of the conjugates vanish modulo x^ℓ.

So, our goal is to show that not more than $\epsilon 2^M$ conjugates vanish modulo x^ℓ. One way to do this is to show that the *product* of all the conjugates does not vanish modulo some larger power of x. Luckily, the product of the 2^M conjugates is a well studied object, and is called the norm of P.[3]

$$\begin{aligned} \text{norm}(P) &= \prod_{\sigma \in \{\pm 1\}^M} P(q_1, \ldots, q_n) \\ &= \prod_{\sigma \in \{\pm 1\}^M} P\left(\sum_{j=1}^{e_1} \sigma_{1j} \sqrt{p_{1j}}, \ldots, \sum_{j=1}^{e_n} \sigma_{nj} \sqrt{p_{nj}} \right) \end{aligned}$$

For example, say that our polynomial is $P(x_1, x_2) = x_1 + x_2$ over the field of three elements, and the irreducible polynomials are $x + 1$ and $x^2 + 1$. The norm of P is:

$$\begin{aligned} &(\sqrt{x+1} + \sqrt{x^2+1})(\sqrt{x+1} - \sqrt{x^2+1}) \\ &(-\sqrt{x+1} + \sqrt{x^2+1})(-\sqrt{x+1} - \sqrt{x^2+1}) \quad = \quad 2(x+1) - 2(x^2+1) \end{aligned}$$

Note that the norm is a *polynomial* over F; all of the square roots cancel out. This is in fact a general phenomenon captured by the following claim which is proved in Appendix E:

Claim 4.4 $\text{norm}(P) \in F[x]$

Lemma 4.2 shows that $\text{norm}(P) \neq 0$ since each element of the product is non-zero. Recall, that our goal is to show that the norm does not vanish modulo some power of x, and since the claim states that the norm of P is in fact a nonzero polynomial over F, all we need to do is to upper bound its degree! We would like to show that the degree of the polynomial can't build up very much over the product of the 2^M conjugates. The problem is that the elements inside the product are *not* polynomials, and it is unclear what their "degree" is.

We solve this problem by defining a degree function $\deg : F[x, \sqrt{p_{11}}, \sqrt{p_{12}}, \ldots, \sqrt{p_n}] \to \mathbb{N}$ with the following three properties:

1. $\deg(fg) = \deg(f) + \deg(g)$
2. $\deg(f + g) = \max(\deg(f), \deg(g))$

[3] This norm is the usual Galois Theory norm over the field extension $F(x)[\sqrt{p_{11}}, \sqrt{p_{12}}, \ldots, \sqrt{p_{ne_n}}]$. Note that we are making implicit use of the fact that the Galois group is $(\mathbb{Z}/2\mathbb{Z})^M$, which follows from Kummer Theory. See Appendix E.

3. If $f \in F[x]$ then $\deg(f)$ is *equal* to the degree of f as a polynomial in x.

In order to define the degree function, we need the following claim which again, is proved in Appendix E:

Claim 4.5 *Every element f in $F[x, \sqrt{p_{11}}, \sqrt{p_{12}}, \ldots, \sqrt{p_{ne_n}}]$ can be* uniquely *represented in the form:*

$$f = \sum_{\alpha} f_\alpha(x) \sqrt{p_{ij}}^{\alpha}$$

where we sum over all α assigning 0 or 1 to each pair (i,j), $f_\alpha(x) \in F[x]$, and $\sqrt{p_{ij}}^{\alpha}$ is an abbreviation for $\prod_{i,j} \sqrt{p_{ij}}^{\alpha_{ij}}$

The degree function for f is defined using this unique representation:

$$\deg(f) = \max_{\alpha} \left(\mathrm{DEG}(f_\alpha) + \sum_{i,j:\alpha_{ij}=1} \frac{\mathrm{DEG}(p_{ij})}{2} \right)$$

Where the max is taken over all non zero summands in the unique representation of f, and DEG is the regular degree function on $F[x]$.

It is a simple matter to verify that this function has the three properties we want from the degree function. We remark that this definition of degree is actually *determined* by the three properties above, because they imply that the degree of $\sqrt{p_{ij}}$ must be half the degree of p_{ij}.

Since $\mathrm{norm}(P) \in F[x]$, we know, by the last property of the degree function, that $\deg(\mathrm{norm}(P))$ is the degree of the norm as a polynomial in $F[x]$. Now, using the other properties of the degree function we have:

$$\begin{aligned}
\deg(\mathrm{norm}(P)) &= \deg\left(\prod_{\sigma \in \{\pm 1\}^M} P\left(\sum_{j=1}^{e_1} \sigma_{1j}\sqrt{p_{1j}}, \ldots, \sum_{j=1}^{e_n} \sigma_{nj}\sqrt{p_{nj}} \right) \right) \\
&= \sum_{\sigma \in \{\pm 1\}^M} \deg\left(P\left(\sum_{j=1}^{e_1} \sigma_{1j}\sqrt{p_{1j}}, \ldots, \sum_{j=1}^{e_n} \sigma_{nj}\sqrt{p_{nj}} \right) \right) \\
&\leq 2^M d \left(\frac{\max(\deg(p_{ij}))}{2} \right)
\end{aligned}$$

Where d is the total degree of P and the max is taken over all p_{ij}'s.

Now, suppose that T conjugates vanish modulo x^ℓ. This means that $\mathrm{norm}(P)$ must vanish modulo $x^{\ell T}$, so it must be true that:

$$\begin{aligned}
\ell T &\leq \deg(\mathrm{norm}(P)) \\
&\leq 2^M d \left(\frac{\max(\deg(p_{ij}))}{2} \right)
\end{aligned}$$

Therefore we have:

$$\frac{T}{2^M} \leq \frac{d \max(\deg(p_{ij}))}{2\ell}$$

The left hand side of the above inequality is just the probability of choosing a "bad" conjugate; that is, one that vanishes modulo x^ℓ. Setting $\ell = \frac{1}{\epsilon}\left(\frac{d \max(\deg(p_{ij}))}{2} \right)$ bounds the probability of error by ϵ.

This concludes the proof of Theorem 4.1. □

5 Characteristic 2

In this section, we sketch an extension of our algorithm to fields of characteristic 2. The essential problem when F is of characteristic 2 is that *no* irreducible polynomials have square roots in $F[[x]]$. Instead, we have to work with *cube roots*. By Lemma 2.1, a polynomial in $F[x]$ has a cube root in $F[[x]]$ iff its constant term is a cube in F. Also, to choose a random conjugate of a cube root, one needs to multiply by a random cube root of unity, rather than ± 1. Thus, for now, we suppose that F contains a primitive cube root of unity ζ. (For finite F of characteristic 2, this is the case iff F is of order 2^k for k even.) Then the algorithm proceeds as follows:

1. **Find Irreducible Polynomials:** Find $\sum_i \log(d_i+1)$ irreducible polynomials p_{ij} ($1 \le i \le n$, $1 \le j \le \log_3(d_i+1)$ whose constant terms are cubes in F.
2. **Approximate Square Roots:** Compute approximations r_{ij} to the cube roots $\sqrt[3]{p_{ij}}$ modulo x^ℓ. This can be done using a method similar to the one for finding approximations to square roots.
3. **Add Randomization:** Set $\overline{\pi}_i = \sum_{ij} \sigma_{ij} r_{ij}$ where σ_{ij} is randomly chosen in $\{1, \zeta, \zeta^2\}$.
4. **Evaluate Polynomial:** Output `'nonzero'` if $P(\overline{\pi}_1, \ldots, \overline{\pi}_n) \neq 0 \pmod{x^\ell}$.

The analysis of this algorithm proceeds much as in the other case, and shows that $\ell = \epsilon^{-1}\mathrm{poly}(d_1, \ldots, d_n)$ is sufficient to obtain error probability ϵ. The number of random bits used by this algorithm is essentially $(\log_2 3)\sum \log_3(d_i+1) = \sum \log_2(d_i+1)$, as before.

The only question that remains is what to do when F does not have a cube root of unity. In the straight-line model, this is easily dealt with: treat ζ as a formally adjoined cube root of 1, reducing ζ^2's to $-\zeta - 1$ when they arise in the computation.

In the black-box model, we treat $P(\overline{\pi}_1, \ldots, \overline{\pi}_n)$ as a bivariate polynomial $g(x, \zeta)$ of degree less than ℓd in x and at most ℓ in ζ. As argued in Appendix A.2, it suffices to substitute $(\ell d) \cdot (\ell + 1)$ values for (x, ζ) to distinguish between the cases that P the zero-polynomial and the case that the real value for $P(\overline{\pi}_1, \ldots, \overline{\pi}_n)$ vanishes modulo x^ℓ. This requires the field to be of size at least $d\ell(\ell+1)$.

Acknowledgements

We are grateful to Madhu Sudan for his comments on this manuscript. We thank Amit Sahai for collaboration at an early stage of this research, and Dan Spielman for useful conversations on this topic. We also thank W. Russell Mann for numerous conversations about algebraic concepts relevant to this work.

References

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the Thirty Third Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of the Thirty Third Annual Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[BCW80] M. Blum, A.K. Chandra, and M.N. Wegman. Equivalence of free Boolean graphs can be tested in polynomial time. *Information Processing Letters*, 10:80–82, 1980.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. In *31st Annual Symposium on Foundations of Computer Science*, volume I, pages 16–25, St. Louis, Missouri, 22–24 October 1990. IEEE.

[BK95] M. Blum and S. Khanna. Designing programs that check their work. *Journal of the Association for Computing Machinery*, 42:269–291, 1995.

[CDGK91] Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, 29 July 1991.

[CK97] Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 200–209, El Paso, Texas, 4–6 May 1997.

[CRS95] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, October 1995.

[Eis95] David Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Graduate Texts in Mathematics 150. Springer-Verlag, 1995.

[GKS90] Dima Yu. Grigoriev, Marek Karpinski, and Michael F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM Journal on Computing*, 19(6):1059–1063, December 1990.

[Kal82] Erich Kaltofen. A polynomial reduction from multivariate to bivariate integral polynomial factorization. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 261–266, San Francisco, California, 5–7 May 1982.

[Kal88] Erich Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the Association for Computing Machinery*, 35(1):231–264, January 1988.

[Lan93] Serge Lang. *Algebra*. Addison-Wesley, 3 edition, 1993.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proofs. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 1–10, 1990.

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[RB91] Ron M. Roth and Gyora M. Benedek. Interpolation and approximation of sparse multivariate polynomials over $GF(2)$. *SIAM Journal on Computing*, 20(2):291–314, April 1991.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, October 1980.

[Sha90] Adi Shamir. IP=PSPACE. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[Str72] V. Strassen. Berechnung und programm I. *Acta Informatica*, 1:320–335, 1972. In German.

[vzGK85] Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, October 1985.

[Zip79] R. E. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of EUROSAM '79*, pages 216–226. Springer-Verlag, 1979. Lecture Notes in Computer Science, vol. 72.

[Zip81] R. Zippel. Newton's iteration and the sparse Hensel algorithm. In *Proceedings of the 1981 ACM Symposium on Symbolic and Algebraic Computation*, pages 68–72, Utah, 1981.

A Two Models for Implicitly Given Polynomials

In this section, we consider two models for considering an implicitly given polynomial and discuss how our algorithm works in each of these settings.

A.1 Straight-Line Programs

Informally, a *straight-line program* [Str72, Kal88] describes a polynomial by a sequence of algebraic operations. More precisely, let D be a ring, $S \subset D$ be a finite set of constants, and $x_1, \ldots, x_n$ be a set of input variables. Then a straight-line program P is a sequence of m statements, where the i'th statement has one of the following forms:

$$y_i \leftarrow x_j \quad \text{for } 1 \leq j \leq n; \qquad y_i \leftarrow s \quad \text{for some } s \in S; \quad \text{or} \qquad y_i \leftarrow \begin{cases} y_j + y_k \\ y_j - y_k \\ y_j \cdot y_k \\ y_j / y_k \end{cases} \quad \text{for some } j, k < i$$

The *output* of P is defined to be y_m. It is clear that every such program defines an easily-computable rational function $P(x_1, \ldots, x_n)$ on D^n. (Assuming arithmetic in D is feasible.) We say that a straight-line program *defines a polynomial*, if the formal expression in the input variables resulting from following the steps of the straight-line program is in fact a polynomial in $x_1, \ldots, x_n$ and if, for every $\alpha \in D^n$ all divisions occurring in the steps of P on input α are actually divisions by invertible elements of D.

Many polynomial functions of interest, such as the determinant, can be expressed as straight-line programs.

Recall that our algorithm requires evaluating a multivariate polynomial $P(x_1, \ldots, x_n)$ defined by a straight-line program at univariate polynomials $\overline{\pi}_1(x), \ldots, \overline{\pi}_n(x)$ modulo x^ℓ. We can do this by simply interpreting the straight-line program for P (over a field F) as a straight-line program over the larger ring $R = F[x]/(x^\ell)$. We need to check two things: evaluating P at points of R^n only results in results in divisions by invertible elements of R, and these divisions can be efficiently. To see this, consider the evaluation of P on $(h_1(x), \ldots, h_n(x)) \in R^n$. Taking every step of this evaluation *modulo* x, it is easy to see that we obtain the evaluation of P on $(a_1, \ldots, a_n) \in F^n$, where $a_1, \ldots, a_n$ are the constant terms of $h_1, \ldots, h_n$. We know that evaluating P on elements of F^n never results in division by 0, so whenever P attempts to invert an element of R, it must be an element with nonzero constant term. The technique in Section 2 for inverting $g_\ell(x)$ during Hensel lifting shows that every element of R with nonzero constant term is invertible and that this inverse can be computed with $\text{poly}(\ell)$ operations in F.

Remark. When discussing straight-line programs over finite fields, there is some ambiguity in the statement $P(x_1, \ldots, x_n) = 0$. It could mean that the polynomial obtained by applying the steps of P to the indeterminates $x_1, \ldots, x_n$ is the *zero element* of the ring $F[x_1, \ldots, x_n]$. Or it could mean that P defines the *zero function* on F^n; that is, $P(a_1, \ldots, a_n) = 0$ for all $a_1, \ldots, a_n \in F$. Although these two conditions are equivalent over infinite fields, they are not in finite fields. For example, the polynomial $x^q - x$ vanishes at all points of $\text{GF}(q)$ but is not the zero element of $\text{GF}(q)[x]$. The two notions are equivalent, however, whenever $|F|$ is greater than the degree d_i of P in each variable x_i. When this condition does not hold, *our algorithms test whether $P(x_1, \ldots, x_n)$ is the zero element of the ring $F[x_1, \ldots, x_n]$*. We note that the Schwartz-Zippel approach requires that the the field is larger than the *total degree* to work at all, whereas our algorithm is meaningful even when the field is $\text{GF}(2)$.

A.2 The Black-Box Model

The definition of this model is as one would expect — instead being given a description of P, our algorithm is given oracle access to a "black-box" that will evaluate P at any point of F^n. In this case, we cannot directly evaluate P at univariate polynomials $\overline{\pi}_1(x), \ldots, \overline{\pi}_n(x)$. Instead, we observe that the univariate polynomial $g(x) = P(\overline{\pi}_1(x), \ldots, \overline{\pi}_n(x))$ has degree less than ℓd, where d is the total degree of P, because each $\overline{\pi}_i(x)$ has degree less than ℓ. Moreover, we can evaluate g at any point of F using the black-box for P. Suppose we evaluate g at ℓd distinct points of F. If all the values obtained are zero, then g must be the zero polynomial so it certainly vanishes modulo x^ℓ and our algorithm should output `'probably zero'`. However, if at least one of the values is nonzero, then P must be a nonzero polynomial and our algorithm should output `'nonzero'`. Note that this approach works whenever $|F| > \ell d$. This restriction on degree is typical of identity-testing algorithms in the black-box model (cf., [CDGK91])

A.3 Other Models

Some other models for representing polynomials considered in the literature are the *dense representation*, which requires that all coefficients are written down; the *sparse representation*, which requires that all nonzero coefficients be written down; and the *formulas*. In the dense and sparse representations, testing whether a polynomial is zero is trivial, and formulas are a special case of a straight-line program.

B Running Time

We analyze the running time of each of the stages of the algorithm. Finding $M = \sum_i \lceil \log(d_i + 1) \rceil$ irreducible polynomials takes time polynomial in M by Lemma 2.2. Extracting square roots by Hensel Lifting modulo x^ℓ takes $\log \ell$ iterations of the algorithm, and each iteration involves $\mathrm{poly}(\ell)$ multiplications and additions of polynomials, where all the operations are done modulo x^ℓ. All this takes time polynomial in ℓ. In Appendix A.1, it was shown how to modify the straight line program to compute the polynomial P, with univariate polynomials substituted in place of its variables, modulo x^ℓ using time polynomial in both ℓ and the length of the program.

Since $\ell = \mathrm{poly}(1/\epsilon, d, M)$ and $M = \mathrm{poly}(d, n)$, the whole running time is polynomial in $1/\epsilon$, d, n, and length of the straight line program.

C Another Algorithm over the Integers

In this section, we mention how the ideas in this paper yield a purely algebraic alternative to Chen and Kao's algorithm over the integers. The main observation, following from a more general form of Hensel's Lemma, is that any prime p that is congruent to 1 modulo 8 has a square root in the 2-adic integers [Eis95, Sec. 7.2]. Moreover, there is a natural notion of approximate solutions in the 2-adics, namely solutions modulo 2^ℓ. Thus our algorithm over $\mathbb{Z}$ and its analysis proceed much as in the finite field case, using the following analogy:

Irreducible polynomials	$\rightsquigarrow$	Prime numbers
F[[x]]	$\rightsquigarrow$	2-adics
Square roots mod x^ℓ	$\rightsquigarrow$	Square roots mod 2^ℓ

The use of the 2-adics is inessential and can be replaced with the q-adics for any fixed prime q.

D Finding Approximations to Square Roots

In this section, we describe how to find approximations to square roots of a polynomial modulo x^ℓ. The method we describe constructs an an approximation modulo $x^{2\ell}$ given an approximation modulo x^ℓ. This is similar to what can be done for Newton approximation.

Say we are trying to approximate the square root of the irreducible polynomial $f(x) \in F[x]$. Let $g_\ell(x)$, $\ell = 1, 2, 3, \ldots$ be successive approximations of $\sqrt{f(x)}$. That is,

$$g_\ell(x)^2 = f(x) \pmod{x^\ell}$$

The first approximation, $g_1(x)$, is simply the square root of f_0 in F: $g_1(x) = \sqrt{f_0}$. (Notice that, in our algorithm, we always construct the polynomial f so that we know the square root of the constant term .)

Now, assume that we have found the ℓ'th approximation, $g_\ell(x)$, such that $g_\ell(x)^2 = f(x) \pmod{x^\ell}$. The 2ℓ'th approximation has the form:

$$g_{2\ell}(x) = x^\ell p(x) + g_\ell(x),$$

where $p(x)$ is a polynomial of degree $\ell - 1$. We want to find a $p(x)$ so that $g_{2\ell}(x)^s = f(x) \pmod{x^{2\ell}}$. Substituting for $g_{2\ell}$, this is equivalent to

$$2x^\ell p(x) g_\ell(x) + g_\ell(x)^2 = f(x) \pmod{x^{2\ell}}$$

Since $g_\ell(x)^2 = f(x) \pmod{x^\ell}$, we know that $f(x) - g_\ell(x)^2$ is divisible by x^ℓ and we obtain:

$$\frac{f(x) - g_\ell(x)^2}{2x^\ell} = p(x) g_\ell(x) \pmod{x^\ell}$$

The polynomial $g_\ell(x)$ has an inverse in $F[[x]]$ which can be found by the following trick. Let $g_\ell(x) = g_0 + x g_\ell^*(x)$, and then note that:

$$\begin{aligned}
\frac{1}{g_\ell(x)} &= \frac{1}{\frac{1}{g_0}\left(1 + \frac{x g_\ell^*(x)}{g_0}\right)} \\
&= g_0 \left(1 - \frac{x g_\ell^*(x)}{g_0} + \frac{x^2 g_\ell^*(x)^2}{g_0^2} - \frac{x^3 g_\ell^*(x)^3}{g_0^3} + \cdots\right)
\end{aligned}$$

Since $p(x)$ has degree $\ell - 1$, we have:

$$p(x) = \left(\frac{f(x) - g_\ell(x)^2}{2x^\ell}\right)\left(\frac{1}{g_\ell(x)}\right) \pmod{x^\ell}$$

So, we can find $p(x)$ by computing $\frac{1}{g_l(x)} \pmod{x^\ell}$ using the trick, and plugging into the above equation.

For example, say that we would like to compute the square root of $f(x) = x + 1$ modulo x^4 in the field of three elements. We obtain the successive approximations $g_1(x) = 1$, and $g_2(x) = 2x + 1$. Now, to obtain $g_4(x)$ we write: $g_4(x) = p(x)x^2 + g_2(x)$ where $p(x)$ has degree 1 and is given by:

$$\begin{aligned}
p(x) &= \left(\frac{f(x) - g_2(x)^2}{2x^2}\right)\left(\frac{1}{g_2(x)}\right) \pmod{x^2} \\
&= \frac{x + 1 - x^2 - x - 1}{2x^2}(1 + x + x^2 + \ldots) \pmod{x^2} \\
&= 1 + x
\end{aligned}$$

So, $g_4(x) = (1 + x)x^2 + 2x + 1 = 1 + 2x + x^2 + x^3$ and it is simple to verify that $g_4(x)^2 = f(x) \pmod{x^4}$.

E Algebraic Lemmas

Proof (of Lemma 2.2): Let L be the (unique) extension of F of degree n. Every nonzero element of L satisfies an irreducible polynomial over F of degree $\leq n$. Each irreducible polynomial of degree $\leq n$ has at most n roots in L. Thus, the number of irreducible polynomials must be at least $|L-\{0\}|/n = (|F|^n-1)/|F|$. □

We now lead up to the proofs of Claims 4.3, 4.4, 4.5. with a few intermediate facts. For notational convenience, let $\{r_1,\ldots,r_M\} = \{p_{ij}: 1 \leq i \leq n, 1 \leq j \leq e_i\}$. Recall that $r_1,\ldots,r_m \in F[x]$ are irreducible polynomials, and we were studying the field extension $K = F(x)[\sqrt{r_1},\ldots,\sqrt{r_n}]$ over $F(x)$. First we obtain the degree and Galois group of this extension using Kummer theory. The following is a special case of [Lan93, VI,Thm. 8.1].

Theorem E.1 *Let B be a subgroup of $F(x)^*$ containing $(F(x)^*)^2$ (i.e., the squares in $F(x)$). Let $K_B = F(x)(\sqrt{B})$ (i.e., adjoin square roots of everything in B.) Then K_B is a Galois extension of $F(x)$ of degree equal to the size of quotient group $B/(F(x)^*)^2$.*

To apply this theorem, let B be the subgroup of $F(x)^*$ generated by $r_1,\ldots,r_M$, along with all the squares in $F(x)^*$. (i.e., $(F(x)^*)^2$) Then $K_B = K$. It is easy to see that every element of B can be written uniquely in the form

$$s \cdot r_1^{\tau_1} \cdots r_M^{\tau_M},$$

where $s \in (F(x)j)^2$ and $\tau_1,\ldots,\tau_M \in \{0,1\}$. Thus, $B/(F(x)^*)^2$ is simply $(\mathbb{Z}/2\mathbb{Z})^M$. So, by Theorem E.1 K is a Galois extension of degree 2^M over $F(x)$. This implies that for each $i \leq M$, $\sqrt{r_i}$ is of degree exactly 2 over $F(x)[\sqrt{r_1},\ldots,\sqrt{r_{i-1}}]$. (If, not K would be of degree strictly less than 2^M over $F(x)$.) We now see that the Galois group of K/F consists exactly of automorphisms σ of the form

$$\sigma(\sqrt{r_i}) = \sigma_i r_i$$

for any $(\sigma_1,\ldots,\sigma_M) \in \{\pm 1\}^M$.

We now proceed to the proofs of Claims 4.3 and 4.4.

Proof (of Claim 4.3): Clearly, it suffices to show that for $i \leq j \leq M$, $\alpha = \sqrt{r_{i+1}} + \cdots + \sqrt{r_j} \in K$ has degree at least 2^{j-i} over L, where $L = F(x)[\sqrt{r_1},\ldots,\sqrt{r_i}]$. Let $f \in L[x]$ be the irreducible polynomial for α over L. For any $\sigma_{i+1},\ldots,\sigma_j \in \{\pm 1\}$, there is an automorphism σ of K fixing L and taking $\sqrt{r_k}$ to $\sigma_k\sqrt{r_k}$ for $i < k \leq j$. (By our description of the Galois group of $K/F(x)$.) Notice that α has 2^{j-i} distinct images under such automorphisms. For any such σ, we have $f(\sigma(\alpha)) = \sigma(f(\alpha)) = 0$, since σ is an automorphism fixing L. Thus, f has at least 2^{j-i} roots, and α is of degree at least 2^{j-i} over L. □

Proof (of Claim 4.4): First observe that by the characterization of the Galois group of $K/F(x)$ above, $\mathrm{norm}(P)$ is in fact the usual Galois-theoretic norm of $P(q_1,\ldots,q_n)$ from K to $F(x)$. And, since norms always lie in the base field (see [Lan93, VI,Thm 5.1]), $\mathrm{norm}(P) \in F(x)$. But we need to prove that $\mathrm{norm}(P)$ is $F[x]$, not $F(x)$.

We need to introduce some terminology. An element of K is called *integral* over $F[x]$ if it is a root of a monic polynomial with coefficients in $F[x]$ (not $F(x)$!). Thus, $\sqrt{r_1},\ldots,\sqrt{r_M}$ are all integral over $F[x]$, as they are roots of the polynomials $Z^2 - r_i$. A standard theorem on ring extensions says that the set of integral elements over a ring form a ring themselves. Thus, $q_1,\ldots,q_n$ are all integral over $F[x]$, and $P(q_1,\ldots,q_n)$ is integral over $F[x]$. Another standard theorem [Lan93, VII, Cor. 1.6] says that if $\beta \in E$ is integral over a ring R then the norm of β from E to the fraction field of R is integral over R. So we see that $\mathrm{norm}(P)$ is actually integral over $F[x]$. Finally, another standard theorem [Lan93, VII, Prop. 1.7] tells us that for

unique factorization domains R, the set of elements of the fraction field of R that are integral over R is R itself. Thus, since $\mathrm{norm}(P)$ is an integral element of $F(x)$, which is the fraction field of $F[x]$, $\mathrm{norm}(P)$ must lie in $F[x]$. □

Proof (of Claim 4.5): Clearly, every element of $F[x, \sqrt{r_1}, \ldots, \sqrt{r_M}]$ can be written in the form

$$\sum_{\alpha} f_\alpha(x) \, (\sqrt{r_1})^{\alpha_1} \cdots (\sqrt{r_M})^{\alpha_M} \, ,$$

where the sum is over all $\alpha \in \{0,1\}^M$ and each $f_\alpha(x) \in F[x]$. Since $F(x)[\sqrt{r_1}, \ldots, \sqrt{r_M}]$ is of degree 2^M over $F(x)$, it must be the case that

$$\{(\sqrt{r_1})^{\alpha_1} \cdots (\sqrt{r_M})^{\alpha_M} : \alpha \in \{0,1\}^M\}$$

is a linearly independent set over $F(x)$. The uniqueness of the representation above follows. □