Deterministic Factorization of Polynomials over Finite Fields

by

David Marquis

A thesis submitted to the Faculty of Graduate and Post Doctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Science

in

Pure Mathematics

Carleton University, Ottawa, Canada

Abstract

We give new results for the problem of deterministically and unconditionally factoring polynomials over finite fields. We make use of composed operations to give new efficient algorithms for the factorization of odd degree polynomials over finite fields. We show that for many finite fields this new algorithm produces factors of a polynomial with higher probability than the previously known algorithms. We remove the assumption of the Extended Riemann Hypothesis from a well known algorithm for factoring polynomials over finite fields in the case that the degree of the polynomial to be factored is coprime to $\phi(p-1)$ where p is the characterisitic of the field and ϕ is the Euler totient function. We also give new results on the factorization of polynomials of bounded degree.

Using the composed product we are also able to give a new proof of a result first given in [29] that a polynomial of degree n over the finite field $\mathbb{F}_p$ can be factored deterministically in a number of operations that is polynomial in n^l where l is the least prime factor of n and $\log(p)$. Use of the composed operations allows us to give a proof that is very concrete compared to how this result has previously been proved.

Acknowledgments

Contents

1 Introduction

In this thesis we are interested in algorithms that efficiently factor polynomials over finite fields. Factoring polynomials over finite fields (FPFF) has applications in computer algebra, coding theory, and cryptography. A *deterministic* algorithm that solves this problem was given by Berlekamp in [5] and [6]. The value of this algorithm is in its generality, any polynomial will eventually be factored by the algorithm, not in its efficiency. Over finite fields of large characteristic the algorithm is very slow as the number of operations needed to factor f over $\mathbb{F}_q$ is not polynomial in $\deg(f)$ and $\log(q)$.

Although an efficient and deterministic algorithm for FPFF does not yet exist there are efficient *probabilistic* algorithms. A probabilistic algorithm is any algorithm that makes random choices. For many computational problems it has proved possible to construct probabilistic algorithms more easily then deterministic ones. Although there has been little progress on solving FPFF deterministically and unconditionally some progress has been made on this problem assuming conjectures such as the Riemann Hypothesis, or generalizations of it like the Extended Riemann Hypothesis, and the Generalized Riemann Hypothesis. We give new results on FPFF that are unconditional and deterministic in Chapter 4 and results that are deterministic under the Generalized Riemann Hypothesis in Chapter 5.

We make precise what we mean by factoring a polynomial.

Definition 1.1. Given a polynomial $f \in \mathbb{F}_q[X]$ then $g \in \mathbb{F}_q[X]$ is a nontrivial factor of f if $g \mid f$ and $0 < \deg(g) < \deg(f)$.

Definition 1.2. To *factor* a polynomial $f \in \mathbb{F}_q$ is to find a nontrivial factor of f.

When a polynomial h is a nontrivial factor of a polynomial f we often abuse notation and simply refer to h as a factor of f.

Definition 1.3. To *completely factor* a polynomial f in $\mathbb{F}_q$ is to find irreducible polynomials $g_1, \ldots, g_k$ for positive integers $e_1, \ldots, e_k$ over $\mathbb{F}_q$ such that

$$f = \prod_{i=1}^{k} {g_i}^{e_i}.$$

For us the problem of factoring polynomials over finite fields (FPFF) means an algorithm which takes a polynomial f and factors it. An algorithm that is guaranteed to find a factor of a generic polynomial, such as Berlekamp's algorithm, can be extended to an algorithm to completely factor the polynomial simply by recursively applying the algorithm to each factor that is found. However, some of the literature on FPFF applies to particular classes of polynomials and has no relation to complete factorization algorithms.

In Chapter 2 we discuss some well known facts which will be used in the rest of the thesis. Many of the results in this chapter deal with the *index* (also known as the discrete log) of a finite field element and a generalization of it that we define for polynomials. The index of a polynomial gives a concrete way to understand many of the known results on FPFF. We also give some well known algorithms for FPFF in this section and some material on graph theory which is helpful (although not strictly necessary) for Chapter 4. In Chapter 3 we survey some of the many results on FPFF. In Chapter 4 we first define five operations on polynomials f and g which factor as $f = \prod_\alpha (X - \alpha)$ and $g = \prod_\beta (X - \beta)$ in an algebraic closure of a finite field. These are

$$f^{(k)} = \prod_\alpha (X - \alpha^k),$$

$$f \oplus g = \prod_\alpha \prod_\beta (X - (\alpha + \beta)), \qquad f \ominus g = \prod_\alpha \prod_\beta (X - (\alpha - \beta)),$$

$$f \otimes g = \prod_\alpha \prod_\beta (X - \alpha\beta), \qquad f \oslash g = \prod_\alpha \prod_\beta (X - \frac{\alpha}{\beta}).$$

These *composed operations* are used to give several new results on FPFF that are deterministic. The results do not make use of any unproven conjectures. The most important new result in this section is an algorithm for factoring polynomials over the finite field $\mathbb{F}_p$ unconditionally and deterministically when $\gcd(n, \phi(p-1)) = 1$. The number of operations used is comparable to a well known result on factoring polynomials deterministically under the Extended Riemann Hypothesis. In Chapter 5 we give results on FPFF that require the assumption of the Extended Riemann Hypothesis and the Generalized Riemann Hypothesis. We use the composed operations as a concrete tool for proving a result given in [29] and [15]. In the last chapter we give some open problems for further investigation.

2 Background

Some background knowledge will be assumed on the part of the reader. The reader should have some familiarity with finite fields and elementary number theory. This material is covered in [21] Chapters 1 and 2. From the theory of computation the reader should be familiar with the basic ideas of the analysis of algorithms and using big-O notation for expressing the number of operations (or running time) of an algorithm. This material is covered near the beginning of [16] or [4].

2.1 Finite Fields Background

We start by giving the background results we need from finite fields and elementary number theory. We use the following notation. Let x be a positive integer. Then $S(x)$ is the largest prime factor of x. Let x and y be positive integers. Then $\nu_y(x)$ is the largest power of y dividing x. For example, if $x = 75$ and $y = 5$ then $\nu_y(x) = 2$. We note that if l is prime $\nu_l(xy) = \nu_l(x) + \nu_l(y)$.

The ring of univariate polynomials over a field $\mathbb{F}_q$ is denoted by $\mathbb{F}_q[X]$. This ring is a unique factorization domain. The finite field $\mathbb{F}_q$ has $q-1$ units and the multiplicative subgroup of its units is denoted $\mathbb{F}_q^*$. Every polynomial in this ring has unique factorization up to multiplication by a unit.

Definition 2.1. A *prime field* is a finite field with a prime number of elements. If the number of elements is p such a field is denoted by $\mathbb{F}_p$.

Theorem 2.2. *Let $\mathbb{F}$ be a finite field of characteristic p. Then $\mathbb{F}$ has p^k elements for some positive integer k.*

Theorem 2.3. *Let p be a prime and n be a positive integer. Every irreducible polynomial in $\mathbb{F}_p[X]$ of degree n divides $X^{p^n} - X$.*

Definition 2.4. Let $\mathbb{F}_q$ be a finite field and $f \in \mathbb{F}_q[X]$. An extension of $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ is called a *splitting field* of f if f splits into a product of linear factors in $\mathbb{F}_{q^k}$ and f has at least one irreducible factor of degree larger than one in any proper subfield of $\mathbb{F}_{q^k}^*$.

Theorem 2.5. *For every prime p and positive integer n, there is a finite field with p^k elements. Furthermore, this finite field is unique up to isomorphism.*

In particular, any finite field with p^k elements is isomorphic to the splitting field $X^{p^k} - X$ over $\mathbb{F}_p$. Let f, g, h be polynomials and $f = g^k h$ where $\gcd(g, h) = 1$ then k is called the *multiplicity* of g. If $k > 1$ then g is a multiple factor of f. A polynomial is called *squarefree* if all its irreducible factors have multiplicity one.

Given a polynomial f of degree n the *reverse* of f denoted f^* is defined

$$f^* = X^n f\big(X^{-1}\big).$$

Theorem 2.6. *Let $\mathbb{F}_{p^k}$ be a finite field, and f be a degree n polynomial over $\mathbb{F}_{p^k}$. If $\gcd(n,p) = 1$ and $g \mid f$ then f' is a polynomial of degree $n-1$ over $\mathbb{F}_{p^k}$ and $g^i \mid f$ if and only if $g^{i-1} \mid f'$.*

Suppose we are given a degree n polynomial f over $\mathbb{F}_{p^k}$ such that $f = g_1 g_2^2 g_3^3 \cdots g_k^k$ for polynomials $g_1, \ldots, g_k$ (possibly equal to 1) and $\gcd(n,p) = 1$. Then for example, $g_2 g_3^2 \cdots g_k^{k-1} = \gcd(f, f')$. Theorem 2.6 is very important for the problem of factoring polynomials. It allows arbitrary polynomials to be decomposed into a product of polynomials that is squarefree. Algorithm 2 in Section 2.2 produces such a decomposition.

Definition 2.7. Let $\mathbb{F}_q$ be a finite field. The *discriminant* of a polynomial $f = (X - r_1) \cdots (X - r_n)$ over $\mathbb{F}_q$ is $\triangle(f) = \prod_{j=1}^{n} \prod_{i=1}^{n} (r_i - r_j)$.

Lemma 2.8. *Let $\mathbb{F}_q$ be a finite field and f be a polynomial over $\mathbb{F}_q$. Then $\triangle(f) = 0$ if and only if f has a factor of multiplicity higher than one.*

For n a positive integer $\phi(n)$ is the *Euler totient function*, the number of positive integers less than n that are coprime to n. If n factors as $n = \prod_{i=1}^{k} p_i^{e_i}$ then

$$\phi(n) = \prod_{i=1}^{k} p_i^{e_i - 1}(p_i - 1).$$

The *order* of x in $\mathbb{F}_q^*$ is the least positive integer such that $x^k = 1$ and is denoted by $\mathrm{ord}(x)$. An element x has order d if and only if

$$x^{d/r} \neq 1$$

for each prime r dividing d. The elements x in $\mathbb{F}_q$ such that $\mathrm{ord}(x) = d$ are *primitive dth roots of unity*. If $d = q - 1$ then x is called a *primitive root*.

Lemma 2.9. *Let $\mathbb{F}_q$ be a finite field, d be a positive integer, and $x \in \mathbb{F}_q^*$. Then*

$$\mathrm{ord}\big(x^d\big) = \frac{\mathrm{ord}(x)}{\gcd(\mathrm{ord}(x), d)}.$$

For example, if g is a primitive root in $\mathbb{F}_q^*$ and l is a prime dividing $q - 1$ then $\mathrm{ord}(g^l) = (q-1)/l$.

Corollary 2.10. *Let $\mathbb{F}_q, x$ be as in Lemma 2.9 and let $s = \nu_l(q-1)$ and $d = (q-1)/l^s$. Then*

$$\nu_l\big(\mathrm{ord}(x)\big) = \nu_l\big(\mathrm{ord}\big(x^d\big)\big).$$

Proof. By Lemma 2.9 we have

$$\nu_l\big(\mathrm{ord}(x)\big) = \nu_l\big(\mathrm{ord}(x)\big) - \nu_l\big(\gcd(\mathrm{ord}(x), d)\big).$$

As no power of the prime l divides d there is also no power of l dividing $\gcd(\mathrm{ord}(x), d)$ so

$$\nu_l\big(\gcd(\mathrm{ord}(x), d)\big) = 0.$$

□

A class of polynomials that is particularly interesting are the cyclotomic polynomials.

Definition 2.11. Let $\mathbb{F}_q$ be a finite field, m a positive integer and k the least positive integer such that $q^k \equiv 1 \pmod{m}$. Let M be the set of all primitive mth roots of unity in $\mathbb{F}_{q^k}^*$. The mth *cyclotomic polynomial* is defined as $\Phi_m = \prod_{\alpha \in M} (X - \alpha)$.

Let ω be a primitive mth root of unity in $\mathbb{F}_{q^k}$ then

$$\Phi_m = \prod_{\substack{i=1 \\ \gcd(i,m)=1}}^{m} \big(X - \omega^i\big).$$

as ω^i is a primitive mth root of unity if and only if i is coprime to m. From this equation it is clear that the degree of Φ_m is equal to $\phi(m)$.

For example, in $\mathbb{F}_q[X]$ the fifth cyclotomic polynomial is

$$\Phi_5 = \frac{X^5 - 1}{X - 1} = X^4 + X^3 + X^2 + X + 1.$$

The proof of Lemma 2.12 is in Chapter 2 of [21].

Lemma 2.12. *Let $\mathbb{F}_q$ be a finite field and m be a positive integer. The coefficients of Φ_m are in $\mathbb{F}_q$.*

Constructing the irreducible factors of cyclotomic polynomials is a problem of central importance in the theory of finite fields. A theorem we will give in the next chapter shows that these polynomials can be factored in deterministic polynomial time assuming a generalization of the Riemann Hypothesis. However, not much is known about factoring these polynomials unconditionally.

The concept of the index of an element will play a central role in the rest of this thesis.

Definition 2.13. Let $x \in \mathbb{F}_q$ and $g \in \mathbb{F}_q$ then the *index* of x denoted $\text{ind}(x)$ with respect to g is the least k such that $g^k = x$ in $\mathbb{F}_q$ assuming such a k exists.

We remark that the index over the finite field $\mathbb{F}_q$ with g, h elements of order d in $\mathbb{F}_q^*$ satisfies

- $\text{ind}_g(1) \equiv 0 \pmod{d}$
- $\text{ind}_g(ab) \equiv \text{ind}_g(a) + \text{ind}_g(b) \pmod{d}$
- $\text{ind}_g(a) \equiv \text{ind}_h(a) \cdot \text{ind}_g(h) \pmod{d}$

for any $a, b \in \mathbb{F}_q^*$. From the first and second properties it is easy to see that $\text{ind}_g(a) + \text{ind}_g(a^{-1}) \equiv 0 \pmod{d}$.

Now we generalize the index of an element to polynomials.

Definition 2.14. Let f be a polynomial in $\mathbb{F}_q[X]$ with roots $r_1, \ldots, r_n$ and $g \in \mathbb{F}_q$ then the *index* of f with respect to g is

$$\text{ind}(f) = \{\text{ind}(r_1), \ldots, \text{ind}(r_n)\}$$

where $\text{ind}(r_1), \ldots, \text{ind}(r_n)$ are computed with respect to g.

For example, let $\mathbb{F}_p$ be a finite field so that $p \equiv 1 \pmod{2^4}$ and $f = X^8 + 1$ (the 16th cyclotomic polynomial) then f factors as

$$f = (X - \omega)(X - \omega^3)(X - \omega^5)(X - \omega^7)(X - \omega^9)(X - \omega^{11})(X - \omega^{13})(X - \omega^{15})$$

if we then compute $\text{ind}(f)$ with respect to ω we have

$$\text{ind}(f) = \{1, 3, 5, 7, 9, 11, 13, 15\}.$$

For $x \in \mathbb{F}_q$ and for l a prime dividing $q - 1$ define $\text{ind}(x)[l^k]$ be the kth digit of $\text{ind}(x)$, computed with respect to some primitive root, expressed in base l with the lowest digit defined as the 0th digit. For example, if $\text{ind}(x) = 13$ then $\text{ind}(x) = 1 + 2^2 + 2^3$ and $\text{ind}(x)[2^0] = 1, \text{ind}(x)[2^1] = 0$.

For f of degree n with roots $r_1, \ldots, r_n$ define

$$\text{ind}(f)[l^k] = \{\text{ind}(r_1)[l^k], \ldots, \text{ind}(r_n)[l^k]\}.$$

There are many possible elements that we could compute $\text{ind}(f)$ with respect to. The next lemma shows that certain equalities between the indexes of elements in $\text{ind}(f)$ are not altered by the choice of element used to compute $\text{ind}(f)$.

Lemma 2.15. *Let a, b be elements in $\mathbb{F}_q^*$, l a prime dividing $q-1$, k a positive integer, and g_1, g_2 elements in $\mathbb{F}_p$ of the same order. For $k < \nu_l(q-1)$ we have* $\text{ind}(a)[l^k] = \text{ind}(b)[l^k]$ *with* ind *computed with respect to g_1 if and only if* $\text{ind}(a)[l^k] = \text{ind}(b)[l^k]$ *with* ind *computed with respect to g_2.*

Proof. There exists some positive integer i so that $g_1^i = g_2$. Suppose i divides $q-1$ then g_2 would not have the same order as g_1 by Lemma 2.9 so i must be coprime to $q-1$. Then if $g_2^k = x$ we have $g_1^{ik} = x$. Let a_1 be $\text{ind}(x)$ computed with respect to g_1, b_1 be $\text{ind}(x)$ computed with respect to g_1, a_2 be $\text{ind}(x)$ computed with respect to g_2, and b_2 be $\text{ind}(x)$ computed with respect to g_2. Then $a_1 = ia_2$ and $b_1 = ib_2$. Hence $\text{ind}(a_1)[l^k] = \text{ind}(b_1)[l^k]$ if and only if $\text{ind}(a_2)[l^k] = \text{ind}(b_2)[l^k]$.

□

For the rest of this thesis we use the following convention. If $\mathbb{F}_q$ is a finite field and x is an element in $\mathbb{F}_q^*$ then $\text{ind}(x)$ is always computed with respect to a primitive root of $\mathbb{F}_q$. We apply Lemma 2.15 in the following way. Let $\mathbb{F}_q$ be a finite field, f be a polynomial of degree n with distinct roots $R = \{r_1, \ldots, r_n\}$, l be a prime dividing $q-1$, $s = \nu_l(q-1)$, and g_1 and g_2 be elements of $\mathbb{F}_q$ of the same order. For each $0 \le k \le l-1$ and $0 \le i \le s-1$ let

$$S_{i,k} = \{r \in R : \text{ind}_{g_1}(r_j)\left[l^i\right] = k\}$$
$$T_{i,k} = \{r \in R : \text{ind}_{g_2}(r_j)\left[l^i\right] = k\}$$

where ind_{g_1} denotes that ind is computed with respect to the element g_1. Then by Lemma 2.15 for any $0 \le i \le s-1$ we have

$$(\#S_{i,0}, \ldots, \#S_{i,l-1}) = (\#T_{i,0}, \ldots, \#T_{i,l-1}).$$

Lemma 2.16 and Lemma 2.17 are about relating the indexes of algebraically related elements. For an element x in a finite field $\mathbb{F}_q$ and a prime l dividing $q-1$ let $s = \nu_2(q-1)$. It is often useful to consider $\text{ind}(x) \equiv a_0 + a_1 l + a_2 l^2 + \cdots + a_{s-1} l^{s-1} \pmod{l^s}$ as an s-tuple. Then

$$\left[\text{ind}(x)\left[l^0\right], \ldots, \text{ind}(x)\left[l^{s-1}\right]\right] = [a_0, \ldots, a_{s-1}]$$

for a_i between 0 and $l-1$. A special case of this Lemma Lemma 2.16 is.

$$\left[\text{ind}(x^l)\left[l^0\right], \ldots, \text{ind}(x^l)\left[l^{s-1}\right]\right] = [0, a_0, \ldots, a_{s-2}].$$

Lemma 2.16. *Let $x \in \mathbb{F}_q^*$, l a prime dividing $q-1$, $s = \nu_l(p-1)$, and $z = \text{ind}(x)[l^i]$ for some $0 < i < s-1$ then for any k such that $k + i \le s-1$ we have $z = \text{ind}(x^{l^k})[l^{k+i}]$.*

Proof. We have $\operatorname{ind}(x) \equiv a_0 + a_1 l + \cdots + a_{s-1} l^{s-1} \pmod{l^s}$ and $\operatorname{ind}(x^{l^k}) = l^k \cdot \operatorname{ind}(x)$. Multiplication by l^k gives

$$l^k \cdot \operatorname{ind}(x) \equiv l^k\big(a_0 + a_1 l + \cdots + a_{s-1} l^{s-1}\big) \equiv a_0 l^k + a_1 l^{1+k} + \cdots + a_{s-1-k} l^{s-1} \pmod{l^s}.$$

Hence $z = \operatorname{ind}(x^{l^k})[l^{k+i}]$.

□

For example, let g be a primitive root modulo a prime p, $x = g^{1+2^2}$, $i = 1$, $k = 1$. Then

$$\begin{aligned} \operatorname{ind}(x)[2^0],\ \operatorname{ind}(x)[2^1],\ \ldots,&\operatorname{ind}(x)[2^2] \ = [1,0,1] \\ \operatorname{ind}(x^2)[2^0],&\operatorname{ind}(x^2)[2^1],\ldots,\operatorname{ind}(x^2)[2^2] = [0,1,0] \end{aligned}$$

Let $z = \operatorname{ind}(x)[2^k]$ then $\operatorname{ind}(x^{2^k})[2^{i+k}] = \operatorname{ind}(x^2)[2^2] = 0 = z$.

Lemma 2.17. *Let $\mathbb{F}_q$ be a finite field, l a prime dividing $q-1$, $s = \nu_l(p-1)$, $x_1, x_2 \in \mathbb{F}_q^*$ such that $x_1^l = x_2^l$. Then for $0 < k < s-2$ we have $\operatorname{ind}(x_1)[l^k] = \operatorname{ind}(x_2)[l^k]$.*

The idea of the proof is that $(x_1/x_2)^l = 1$ in $\mathbb{F}_q$ so $x_1 = \omega x_2$ for ω an lth root of unity. Then $\operatorname{ind}(x_1) = \operatorname{ind}(\omega) + \operatorname{ind}(x_2) \equiv \operatorname{ind}(x_2) + k l^{s-1}$ for some k with $0 < k < l$.

Lemma 2.17 is relevant to factorization of polynomials of the form $X^l - a$ for $a \in \mathbb{F}_q^*$. For example, if r_1, r_2 are roots of $X^l - a$ for $a \in \mathbb{F}_q^*$ then for $0 < k < s-2$ we have $\operatorname{ind}(x_1)[l^k] = \operatorname{ind}(x_2)[l^k]$.

Lemma 2.18. *Let $\mathbb{F}_q$ be a finite field and $x \in \mathbb{F}_q^*$ then*

- $\operatorname{ord}(x) = \frac{q-1}{\gcd(\operatorname{ind}(x), q-1)}$;
- *Let l be a prime dividing $q-1$, $s = \nu_l(q-1)$, $\operatorname{ind}(x) \equiv a_0 + a_1 l + \ldots + a_{s-1} l^{s-1} \pmod{l^s}$, and t be the largest positive integer so that $a_0, \ldots, a_t$ are all zero. Then $\nu_l(\operatorname{ord}(x)) = s - t$.*

Proof. For the first result let g be an arbitrary primitive root. Then $x = g^{\operatorname{ind}_g(x)}$. Lemma 2.9 gives that $\operatorname{ord}(x) = \operatorname{ord}(g)/\gcd(\operatorname{ord}(g), d)$. The result follows by taking $d = \operatorname{ind}(x)$. For the second result

$$\begin{aligned} \nu_l\big(\operatorname{ord}(x)\big) &= \nu_l\big(\operatorname{ord}(g)\big) - \nu_l\big(\gcd(\operatorname{ind}(x), \operatorname{ord}(g))\big) \\ &= \nu_l(q-1) - \nu_l\big(\gcd(\operatorname{ind}(x), q-1)\big) \\ &= s - \nu_l\big(\gcd(\operatorname{ind}(x), q-1)\big) \end{aligned}$$

and we have that l^t is the largest power of l dividing $\operatorname{ind}(x)$ so $\gcd(\operatorname{ind}(x), q-1) = l^t$. □

Computing the index of $x \in \mathbb{F}_q$ is also called the *discrete log* computation. There is no polynomial time algorithm for computing either the index or the order of arbitrary elements of $\mathbb{F}_q$. However, some information about the index of a polynomial can be inferred without computing it directly. For example, consider a polynomial f over $\mathbb{F}_p$ which is a product of linear factors and which satisfies $f = h(X^2)$ for some polynomial h. Then f factors as

$$\begin{aligned} f &= (X^2 - r_1^2)(X - r_2^2)\cdots(X - r_k^2) \\ &= (X - r_1)(X + r_1)(X - r_2)(X + r_2)\cdots(X - r_k)(X + r_k) \end{aligned}$$

for some roots $r_1, \ldots, r_k$. Let g be a generator of $\mathbb{F}_p$. Then $-1 = g^{(p-1)/2}$. Then for each r_i there exists an e_i so that $g^{e_i} \equiv r_i \pmod{p}$ and hence

$$f = \big(X - g^{e_1}\big)\big(X + g^{e_1+(p-1)/2}\big)\cdots\big(X - g^{e_k}\big)\big(X + g^{e_k+(p-1)/2}\big).$$

Now let $s = \nu_2(p-1)$ then $(p-1)/2 = j2^{s-1}$ for some odd j with $j \equiv a_0 + a_1 2 + \cdots + a_{s-1}2^{s-1} \pmod{2^s}$. Hence

$$\begin{aligned} \operatorname{ind}(r_1) &= e_1 + (p-1)/2 = e_1 + \big(a_0 + a_1 2 + \cdots + a_{s-1}2^{s-1}\big)2^{s-1} \\ &\equiv e_1 + a_0 2^{s-1} \pmod{2^s}. \end{aligned}$$

Consider $\operatorname{ind}(r_1) - \operatorname{ind}(-r_1) \equiv a_0 2^{s-1} \pmod{2^s}$. Hence $\operatorname{ind}(r_1)\big[2^{s-1}\big] \neq \operatorname{ind}(-r_1)\big[2^{s-1}\big]$. We have found some information about the bits of $\operatorname{ind}(f)$ when f is equal to $h(X^2)$ for some polynomial h.

Definition 2.19. Let $x \in \mathbb{F}_q^*$ and m be a divisor of $q-1$. Then x is an mth *residue* if $\operatorname{ind}(x) \equiv 0 \pmod{m}$ (with the index computed with respect to some primitive root g). Otherwise x is an mth *nonresidue*. If $m = 2$ then x is also called a *quadratic residue*.

Let g, h be primitive roots modulo p. By Lemma 2.18 $\operatorname{ord}(g) = (q-1)/\gcd(\operatorname{ind}_h(g), q-1)$ hence $\gcd(\operatorname{ind}_h(g), q-1) = 1$. In particular,

$$\operatorname{ind}_h(g) \not\equiv 0 \pmod{l}$$

for any l dividing $q-1$. Recall the "change of basis" formula $\operatorname{ind}_g(a) \equiv \operatorname{ind}_h(a) \cdot \operatorname{ind}_g(h) \pmod{q-1}$. This implies that

$$\operatorname{ind}_g(a) \equiv \operatorname{ind}_h(a) \cdot \operatorname{ind}_g(h) \pmod{l}$$

for any prime l dividing $q-1$. If a is an lth residue with $\mathrm{ind}(a)$ computed with respect to g then $\mathrm{ind}_g(a) \equiv 0 \pmod{l}$. As $\mathrm{ind}_g(h) \not\equiv 0 \pmod{l}$ it follows that $\mathrm{ind}_h(a) \equiv 0 \pmod{l}$. Hence a is a lth residue regardless of which primitive root is used to compute $\mathrm{ind}(a)$.

The next theorem is important because it shows that the property of being an mth residue can be tested in polynomial time.

Theorem 2.20. *Let $a \in \mathbb{F}_q^*$ and l be a divisor of $q-1$. Then*

- *a is a lth residue in $\mathbb{F}_q^*$ if and only if $a^{(q-1)/l} = 1$;*
- *the polynomial $X^l - a$ has a root in $\mathbb{F}_q^*$ if and only if a is an lth residue.*

Proof. Suppose a is a lth residue, and g is a primitive root of unity modulo p. There exists a positive integer k so that $a = g^{lk}$. Let $b = g^k$ then $a^{(q-1)/l} = (b^l)^{(q-1)/l} = 1$ by Fermat's little theorem.

Suppose $a^{(q-1)/l} = 1$ then there exists an integer i so that $g^{i(q-1)/l} = 1$. Let $s = \nu_l(q-1)$ then $q - 1 = kl^s$ for some k that is not divisible by k. We have

$$g^{i(q-1)l^{-1}} = g^{i(q-1)l^{-1}} = g^{ikl^{s-1}} = 1$$

and so the $\mathrm{ord}(g) = q-1$ divides ikl^{s-1}. Hence i is divisible by l so a is an lth residue. This shows the first statement.

Suppose a is a lth residue, and g is a primitive root of unity modulo p. There exists a positive integer k so that $a = g^{lk}$. Then g^k is a root of $X^l - a$. In the other direction suppose $X^l - a$ has a root z. Then $z = g^i$ for some i. Hence

$$\mathrm{ind}\big(a\big) = \mathrm{ind}\big((g^i)^l\big) = l \cdot \mathrm{ind}\big(g^i\big) \equiv 0 \pmod{l}.$$

□

Lemma 2.21. *Let q be a prime power and l be a divisor of $q-1$. The number of lth residues in $\mathbb{F}_q^*$ is $(q-1)/l$.*

Proof. Any lth residue is a root of $g = X^{(q-1)/l} - 1$ by Theorem 2.20 and g is squarefree by Theorem 2.6. Let

$$h = \prod_{i=1}^{l-1} (X^{(q-1)/l} - \omega^i).$$

We have the factorization $X^{q-1} - 1 = gh$. Hence $g \mid X^{q-1} - 1$ and so g has $(q-1)/l$ distinct roots in $\mathbb{F}_q$. □

Corollary 2.22. *Let q be a prime power and l be a divisor of $q-1$. A random element of $\mathbb{F}_q^*$ is an lth residue with probability $1/l$.*

Lemma 2.23. *Let $\mathbb{F}_q$ be a finite field. Then $\alpha \in \mathbb{F}_q$ has order dividing d in $\mathbb{F}_q$ if and only if α is a root of $X^d - 1$ and we have the factorization. $X^d - 1 = \prod_{\mathrm{ord}(\alpha)|d} (X - \alpha)$.*

With $d = \frac{p-1}{2}$ this Lemma 2.23 is used for polynomial factorization in [5] and [9]. As an example of Lemma 2.23 if x is a quadratic residue then $\mathrm{ord}(x)$ divides $(p-1)/2$ and so x is a root of $X^{(p-1)/2} - 1$.

In Theorem 2.31 we will see how to reduce factorization of an arbitrary polynomial over the finite field $\mathbb{F}_{p^k}$ to factoring polynomials that are squarefree products of linear factors over $\mathbb{F}_p$. Hence this class of polynomials is particularly important in the study of deterministic algorithms for FPFF. We note that this reduction is too slow to be of use for probabilistic FPFF algorithms. As this class is used so often we define a subset L_n of $\mathbb{F}_q^*[X]$ as

$$L_n = \left\{ f : f = \prod_{i=1}^{n} (X - a_i), a_i \in \mathbb{F}_q^* \right\}.$$

The number of polynomials in L_n is $(q-1)^n$ since there are $q-1$ choices for each root.

Many algorithms for FPFF work by finding an element that is not a unit in the ring $R = \mathbb{F}_q[X]/(f)$. If α is such an element and is nonzero then $\gcd(\alpha, f)$ is a nontrivial factor of f. If f factors as $f = f_1^{e_1} \cdots f_k^{e_k}$ then the ring R decomposes as a product of finite fields as

$$R = \mathbb{F}_q[X]/(f) \cong \mathbb{F}_q[X]/(f_1^{e_1}) \oplus \cdots \oplus \mathbb{F}_q[X]/(f_k^{e_k}).$$

Any element α of this ring can be expressed as a vector $(a_1, \ldots, a_k)$ with $a_i = \alpha \pmod{f_i^{e_i}}$. The a_i are referred to as the *components* of α.

Elements of the form X^k in R often play a role in the algorithms discussed in this thesis because their components are powers of the roots of f. In particular, if f is a polynomial in L_n over a finite field $\mathbb{F}_q$ and r is a root of f then r^k is a component of X^k in R.

Given an element α in R with f a polynomial in L_n we sometimes need to compute the polynomial that has the components of α as its roots. Let $\mathbb{F}_q$ be a finite field, f be a squarefree polynomial in L_n with roots $r_1, \ldots, r_n$, $R = \mathbb{F}_q[X]/(f)$, and $\alpha \in R$. The element α is a polynomial of degree less than n and has a tuple of components $(\alpha_1, \ldots, \alpha_n)$ with α_i equal to α evaluated at r_i. The *minimal polynomial* of α is defined

$$M_X(\alpha) = (X - \alpha_1) \cdots (X - \alpha_n).$$

The *Berlekamp subalgebra* B is the subring of $R = \mathbb{F}_q[X]/(f)$ such that an element of R is in B if and

only if all its components are in $\mathbb{F}_q$. We note that over the finite field $\mathbb{F}_{p^k}$ the Berlekamp subalgebra is defined by some authors as the elements of R with all their components in $\mathbb{F}_p$. The Berlekamp subalgebra is used in many results on factoring polynomials over finite fields. The next theorem is a useful equivalent characterization of the Berlekamp subalgebra.

Theorem 2.24. *Let $\mathbb{F}_q$ be a finite field and $a \in \mathbb{F}_q[X]/(f)$ then a is an element of the Berlekamp subalgebra if and only if $a^q = a$.*

In the remainder of this section we discuss resultants. Resultants will be one of our main tools in the new algorithms we give in Chapter 4 and Chapter 5.

Let $\mathbb{F}$ be a field and $f = \sum_{i=0}^{n} a_i X^i$ for some $a_0, \ldots, a_n$ in $\mathbb{F}$ and $g = \sum_{i=0}^{m} b_i X^i$ for some $b_0, \ldots, b_m$ in $\mathbb{F}$. The *Sylvester matrix* is defined

$$M = \begin{pmatrix} a_0 & & & & b_0 & & & \\ a_1 & a_0 & & & b_1 & b_0 & & \\ a_2 & a_1 & \ddots & & b_2 & b_1 & \ddots & \\ \vdots & a_2 & \ddots & a_0 & \vdots & b_2 & \ddots & b_0 \\ a_n & \vdots & \ddots & a_1 & b_m & \vdots & \ddots & b_1 \\ & a_n & & \vdots & & b_m & & \vdots \\ & & & a_n & & & & b_m \end{pmatrix}$$

where the empty positions are filled with zeroes. The Sylvester matrix is illustrated by the following example for two polynomials of degree 5 and 3.

$$M = \begin{pmatrix} a_0 & 0 & 0 & b_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & b_1 & b_0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & b_2 & b_1 & b_0 & 0 & 0 \\ a_3 & a_2 & a_1 & b_3 & b_2 & b_1 & b_0 & 0 \\ a_4 & a_3 & a_2 & 0 & b_3 & b_2 & b_1 & b_0 \\ a_5 & a_4 & a_3 & 0 & 0 & b_3 & b_2 & b_1 \\ 0 & a_5 & a_4 & 0 & 0 & 0 & b_3 & b_2 \\ 0 & 0 & a_5 & 0 & 0 & 0 & 0 & b_3 \end{pmatrix}.$$

Definition 2.25. Let R be a unique factorization domain with field of fractions F, $f, g \in R[X]$ monic nonzero polynomials of degrees n and m, respectively, and $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ the roots of f and g, respectively, in an extension of F, counted with multiplicities

The *resultant* of f and g with respect to the indeterminant X is defined as

$$\mathrm{Res}_X(f,g) = \prod_{i=1}^{n}\prod_{j=1}^{n}(\alpha_i - \beta_j)$$

This definition follows [16] Chapter 6. For example, let $f = X - Y$ and $g = X - Y^2$ then $\mathrm{Res}(f, g, X) = Y - Y^2$.

The most useful property of the resultant is that it can be computed on polynomials without given factorizations. It can be computed as the determinant of the Sylvester matrix or using a variation of the Euclidean algorithm [16].

Theorem 2.26. *Let $\mathbb{F}$ be a field and $f = \sum_{i=0}^{n} a_i X^i$ for some $a_0, \ldots, a_n$ in $\mathbb{F}$ and $g = \sum_{i=0}^{m} b_i X^i$ for some $b_0, \ldots, b_m$ in $\mathbb{F}$ with Sylvester matrix M. Then*

$$\mathrm{Res}_X(f,g) = \det(M)$$

2.2 Algorithmic Background

The running times of algorithms in this thesis are given in terms of $\mathbb{F}_p$-*operations*. An $\mathbb{F}_p$-operation is one of the operations $(+, -, *, \div)$ in the finite field $\mathbb{F}_p$ where p is a prime. For the rest of this thesis, whenever we refer to the number of operations of an algorithm we mean the number of $\mathbb{F}_p$-operations.

It is easiest to compute the running time of an algorithm which has input in the finite field $\mathbb{F}_{p^k}$ in two steps: first, compute the number of $\mathbb{F}_{p^k}$ operations that are used. We can take $\mathbb{F}_{p^k} = \mathbb{F}_p[X]/(f)$ for an irreducible polynomial f of degree k. The elements of $\mathbb{F}_{p^k}$ can be treated as polynomials of degree less than k. The operations $(+, -, *, \div)$ on two elements of $\mathbb{F}_{p^k}$ can then be performed in $O(k)$ $\mathbb{F}_p$-operations for $+$ and $-$ and $O(k^2)$ $\mathbb{F}_p$-operations for $*$ and $\div$ by performing the addition, subtraction, multiplication, or division of two elements of $\mathbb{F}_{p^k}$ in the straightforward way and then computing the remainder of the result after being divided by f. Hence an algorithm that uses $O(L)$ $\mathbb{F}_{p^k}$-operations uses $O(Lk^2)$ $\mathbb{F}_p$-operations.

We note that another way to state running times of finite field algorithms is in terms of *bit operations*. A bit operation is a logical XOR or AND operation on two bits. For example, [4] uses this style. The four $\mathbb{F}_p$ operations given above can be computed in $O(\log(p))$ bit operations for $+$ and $-$, and $O(\log(p)^2)$ bit operations for $*$ and $\div$.

An algorithm f uses $\mathrm{poly}(a_1, \ldots, a_k)$ operations if there is some polynomial upper bound in the parameters $a_1, \ldots, a_k$ on the number of operations. For example, Berlekamp's algorithm could be said to use $\mathrm{poly}(q, n)$ operations.

In this thesis an algorithm is called *polynomial time* if the number of $\mathbb{F}_p$-operations used by the algorithm is polynomial in the length of the input. For example, to store a polynomial of degree n over $\mathbb{F}_p$ requires storing the $n+1$ coefficients which are integers between 0 and $p-1$. An integer in this range could have $O(\log(p))$ bits so the input size for the whole polynomial is $O(n\log(p))$. Similarly, for the finite field $\mathbb{F}_q$ a polynomial has a length of $O(n\log(q))$ bits. Berlekamp's algorithm requires $O(q)$ gcd computations of elements of $\mathbb{F}_q[X]$ in the worst case so the number of operations used is proportional to q. There is no polynomial function of $O(n\log(q))$ that could upper bound q so this algorithm is not polynomial time.

In the next chapter some results are stated using soft-O notation. The expression $f(n) = \tilde{O}(g(n))$ is shorthand for

$$f(n) = O\Big(g(n)\log\big(g(n)\big)^k\Big)$$

for any positive integer k.

Lemma 2.27. *Let $\mathbb{F}_{p^k}$ be a finite field, $q = p^k$, l a prime dividing $q-1$, and $s = \nu_l(q-1)$. Then given an lth nonresidue we can construct an l^s primitive root of unity in $O(\log(q)k^2)$ operations.*

Proof. We have that $m = (q-1)/l^s$ is not divisible by l. Let x be an lth nonresidue and $y = x^m$. Then $\operatorname{ind}(x) \equiv a_0 + a_1 l + a_2 l^2 + \cdots + a_{s-1}l^{s-1} \pmod{l^s}$ where $0 \le a_i < l$ and $a_0 \neq 0$. Then

$$\operatorname{ind}(y) \equiv m\big(a_0 + a_1 l + a_2 l^2 + \cdots + a_{s-1}l^{s-1}\big) \equiv ma_0 \neq 0 \pmod{l}$$

so y is an lth nonresidue and hence the order of y is divisible by a power of l^s. On the other hand for any prime r dividing $p-1$ we have that $\operatorname{ind}(x) \equiv 0 \pmod{r}$. Hence the order of y is a power of l and so y must be an l^s primitive root of unity.

We now consider the number of operations. Suppose $\mathbb{F}_q$ is given as $\mathbb{F}_p[X]/(g)$ for some polynomial g of degree k. Computation of y takes $O(\log(q))$ multiplications in $\mathbb{F}_q$ using a simple repeated squaring algorithm. Each multiplication can be performed as the multiplication of two polynomials of degree less than or equal to k followed by a division by g. Both multiplication and division of polynomials of degree less than or equal to k take $O(k^2)$ operations. □

When attempting to factor a polynomial efficiently over a finite field $\mathbb{F}_q$ it may be useful to use fast algorithms for polynomial multiplication such as Karatsuba or FFT based algorithms. However, since the problem of *deterministically* factoring polynomials over finite fields is of theoretical interest, to make expressions for the number of operations of an algorithm simpler and more concise we will assume a straightforward algorithm for polynomial multiplication is used when considering the number

of operations of our algorithms.

Theorem 2.28. *Let $\mathbb{F}_{p^k}$ be a finite field.*

- *Multiplication and division of two polynomials in $\mathbb{F}_{p^k}[X]$ of degree at most n can be performed using $O((nk)^2)$ $\mathbb{F}_p$-operations.*
- *Let f and g be polynomials in $\mathbb{F}_{p^k}[X]$ of degree at most n. Then $f \pmod g$ can be computed using $O((nk)^2)$ $\mathbb{F}_p$-operations.*
- *The greatest common divisor of two polynomials in $\mathbb{F}_{p^k}[X]$ of degree at most n can be performed using $O((nk)^2)$ $\mathbb{F}_p$-operations.*

Earlier we saw that if a polynomial f factors as $f = f_1^{e_1} \cdots f_k^{e_k}$ over the finite field $\mathbb{F}_q$ then the ring $\mathbb{F}_q[X]/(f)$ decomposes as a product of finite fields

$$\mathbb{F}_q[X]/(f) \cong \mathbb{F}_q[X]/(f_1^{e_1}) \oplus \cdots \oplus \mathbb{F}_q[X]/(f_k^{e_k}).$$

We now consider the number of operations needed to compute in this ring. We recall that if an element α in $\mathbb{F}_q[X]/(f)$ is not a unit then $\gcd(\alpha, f)$ is a factor of f.

Theorem 2.29. *Let f be a nonzero polynomial in $\mathbb{F}_{p^k}[X]$ of degree n, $R = \mathbb{F}_{p^k}[X]/(f)$.*

- *Addition and subtraction of elements in R can be performed using $O(nk)$ $\mathbb{F}_p$-operations.*
- *Multiplication of elements in R can be performed using $O((nk)^2)$ $\mathbb{F}_p$-operations.*
- *Inversion of elements in R^* can be performed using $O((nk)^2)$ $\mathbb{F}_p$-operations.*
- *Exponentiation of elements in R to the power e can be performed using $O(\log(e)(nk)^2)$ $\mathbb{F}_p$-operations.*

In some results on FPFF it is necessary to find a basis for the Berlekamp subalgebra. To construct a basis of the Berlekamp subalgebra of $R = \mathbb{F}_q[X]/(f)$ where f is a polynomial of degree n we form a matrix M with rows $m_1, \ldots, m_n$ with m_i defined

$$m_i = (a_0, \ldots, a_{n-1})$$

for $a_j \in \mathbb{F}_q$ such that

$$\left(X^i\right)^q = \sum_{i=0}^{n-1} a_i X^i$$

(computing $(X^i)^q$ is done in R). Then we find the kernel of $M - I$. For example, a basis for the Berlekamp

subalgebra of $\mathbb{F}_{19}[X]/(f)$ with $f = X^4 + X^3 + X^2 + X + 1$ is found by computing

$$\begin{aligned} X^p &= \left(X^5\right)^3 X^4 = X^4 \ = -\left(X^3 + X^2 + X + 1\right) \\ X^{2p} &= \left(X^p\right)^2 = \left(X^4\right)^2 = X^3 \\ X^{3p} &= \left(X^p\right)^3 = \left(X^4\right)^3 = X^2 \end{aligned}$$

using the fact that $X^5 = 1$. Then the matrix is

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

A basis for the kernel of $M - I$ is $\{[-2 \ 0 \ 1 \ 1]^T, [-2 \ 1 \ 0 \ 0]^T\}$ so any element in the span of these vectors is in the Berlekamp subalgebra.

Theorem 2.30. *Let $\mathbb{F}_{p^k}$ be a finite field, f a degree n polynomial in $\mathbb{F}_{p^k}[X]$, and $R = \mathbb{F}_{p^k}[X]/(f)$, and let B denote the Berlekamp algebra of R considered as a $\mathbb{F}_{p^k}$-vector space. A basis for B can be found using $O((\log(q) + nk)n^2k^2)$ operations.*

Proof. There are two computational steps that are involved. The first is constructing the set of powers $X^q, (X^q)^2, \ldots, (X^q)^{n-1}$. One way to perform this calculation is to compute X^q modulo f and then get the rest of the powers by performing n multiplications. These steps take $O(\log(q)(nk)^2)$ and $n(nk)^2$ operations respectively by Theorem 2.29. The next step involves finding the kernel of the $M - I$. This can be done using the elementary operations of adding rows of the matrix, multiplying rows of the matrix by elements of $\mathbb{F}_q$ and swapping rows of the matrix to perform Gaussian elimination. Elimination takes $O((nk)^3)$ operations. □

We note that when f splits into a product of linear factors the Berlekamp subalgebra of $R = \mathbb{F}_q[X]/(f)$ is equal to R.

We now discuss some of the algorithms used for factoring polynomials over finite fields. Many algorithms for polynomial factorization proceed in three stages.

- Squarefree Factorization: The squarefree factors of f are found.
- Distinct Degree Factorization: The squarefree factors of f are factored in products of irreducible polynomials of equal degree.
- Equal Degree Factorization: The distinct degree factors of f are factored into irreducibles.

Algorithm 1 is a squarefree factorization algorithm.

Algorithm 1: SQF_DECOMP

Input: $\mathbb{F}_q, f$
$\mathbb{F}_q$ a finite field
f a polynomial in $\mathbb{F}_q[X]$ with factorization $f = \prod_{i=1}^{k} f_i^{e_i}$ with $0 < e_1 < e_2 < \cdots < e_k$
Output: a list of pairs of the form $[f_i, e_i]$ for $i = 1, \ldots, k$

$g = \gcd(f, f')$
if $f' = 0$ **then**
    find h so that $h^p = f$
    $[[h_1, e_1], \ldots, [h_s, e_s]] =$ SQF_DECOMP(h)
end
else
    if $g = 1$ **then**
        Return $[[f, 1]]$
    end
    else
        Return [SQF_DECOMP(g),SQF_DECOMP(f/g)]
    end
end

Algorithm 2 is a distinct degree factorization algorithm. It makes use of Theorem 2.3. From this theorem we have that every irreducible polynomial of degree n over the finite field $\mathbb{F}_q$ divides the polynomial $X^{q^n} - X$. In Algorithm 2 given a finite field $\mathbb{F}_q$ we define

$$\tau(x) = x^q$$

for $x \in \mathbb{F}_q[X]/(f)$.

Algorithm 2: DIS_DEG

Input: $\mathbb{F}_q, f$
$\mathbb{F}_q$ a finite field
f a squarefree polynomial in $\mathbb{F}_q[X]$
Output: a list M of factors of f in $\mathbb{F}_q$ such that each factor is a product of irreducible polynomials of equal degree

$g = X$
Initialize M as an empty tuple
for $i = 1 \ldots \deg(f)$ **do**
    $g = \tau(g)$
    $f_i = \gcd(g - X, f)$
    Add f_i to M
    $f = f/f_i$
end
Return M

Our interest in this thesis is purely in deterministic algorithms for FPFF. We give Algorithm 3 as an example of a *probabilistic* algorithm for equal degree factorization. This algorithm is polynomial time.

The main idea of the algorithm is that for any element α in the Berlekamp subalgebra of $\mathbb{F}_q[X]/(f)$ we have $\alpha^q = (\alpha, \ldots, \alpha)$ so $\alpha^{q-1/2} = (\pm 1, \pm 1, \ldots, \pm 1)$. Hence it is likely that the vector $\alpha^{q-1/2}$ is not equal to all 1s or -1s.

Algorithm 3: DIS_DEG

Input: $\mathbb{F}_q, f$
  $\mathbb{F}_q$ a finite field
  f a squarefree polynomial in $\mathbb{F}_q[X]$
Output: A nontrivial factor of f

Contruct a random α in the Berlekamp subalgebra of $\mathbb{F}_q[X]/(f)$
$g = \gcd(f, \alpha)$
if $0 < \deg(g) < \deg(f)$ **then**
    `/* check that α does not have any components that are zero */`
    Return g
end
$\beta = \alpha^{(q-1)/2}$
$g = \gcd(\beta - 1, f)$
if $0 < \deg(g) < \deg(f)$ **then**
    Return g
end
else
    Return FAIL;
end

The algorithm is not guaranteed to produce a factor but if it returns failure it can be repeated until a factor is discovered. We note that although Algorithm 3 is given as an example of an equal degree factorization algorithm, this algorithm will produce a factor of any squarefree polynomial.

Due to the fact that Algorithm 1, and Algorithm 2 are deterministic polynomial time algorithms, when we are considering the problem of *deterministic* polynomial factorization we may as well assume that the polynomials we wish to factor are squarefree products of irreducible polynomials of the same degree.

The next theorem reduces the problem of factoring polynomials over $\mathbb{F}_q$ to finding roots of polynomials of over $\mathbb{F}_p$. This theorem is too slow to be used by probabilistic algorithms for factoring polynomials but it is a useful simplification for factoring polynomials deterministically.

Theorem 2.31. *Let p be a prime, $q = p^k$, f a polynomial of degree n in $\mathbb{F}_q[X]$, and assume f has i irreducible factors for $i \geq 2$. Then there are a pair of polynomials $g \in \mathbb{F}_q[X]$ and $h \in \mathbb{F}_p[X]$ with the following properties*

- *h splits completely in $\mathbb{F}_p[X]$;*
- *for any root a of h in $\mathbb{F}_p$, $\gcd(g - a, f)$ is a nontrivial factor of f;*
- $\deg(g) < n$;
- $\deg(h) = i$.

These polynomials can be computed using $O((nk)^3)$ $\mathbb{F}_p$-operations.

The idea of the proof is that given f a polynomial over $\mathbb{F}_q$ in the indeterminant X we should choose an element $g(X)$ in Berlekamp subalgebra generated by f. Then there exists some $c \in \mathbb{F}_p$ so that $g - c$ and f have a common factor. Recalling the earlier definition of the resultant we have the values of c such that $\gcd(f, g - c) = 0$ are the roots of the polynomial $h(Y) = \mathrm{Res}(f, g - Y)$ where the resultant is computed with respect to X. To factor f over $\mathbb{F}_q$ it suffices to find a root of h over $\mathbb{F}_q$. Root finding over $\mathbb{F}_q$ may be reduced to root finding over $\mathbb{F}_p$ as is explained in [21] Chapter 4.

Much of the remainder of this chapter will be spent on two algorithms for factoring polynomials in L_n (products of linear factors of degree n) over a finite field $\mathbb{F}_q$. These are Algorithm 4 and Algorithm 6. We also consider computing a type of congruence involving $\mathrm{ind}(x)$ for x an element of a finite field. We connect this to a special case of factoring polynomials over finite fields; finding roots of polynomials of the form $X^n - a$ for $a \in \mathbb{F}_q$.

Algorithm 4 factors any polynomial which has two roots of distinct order. It is interesting because it is deterministic and is not conditional on any unproven conjectures. However, it requires a list of prime factors of $q - 1$ which cannot be computed in deterministic polynomial time in $\log(q)$. It is also difficult to know in advance whether a polynomial does have roots of distinct order. Algorithm 4 is easily derived from known results however it does not seem to have been stated explicitly before.

Algorithm 4: DIFF_ORDS

Input: $\mathbb{F}_q, F, f$
  $\mathbb{F}_q$ a finite field
  F a list of the prime factors of $q - 1$
  $f \in L_n$ over $\mathbb{F}_q$ such that f is squarefree
Output: A factor of f

let $R = \mathbb{F}_q[X]/(f)$
for $l \in F$ **do**
  $s = \nu_l(q - 1)$
  $d = (q - 1)/l^s$
  /* note that α is an element of R so X^d is reduced modulo f */

  $\alpha = X^d$
  /* note that $\alpha^{l^s} = 1$ so the following while loop terminates after s iterations */
  while $\alpha \notin \mathbb{F}_q$ **do**
    $\beta = \alpha$
    $\alpha = \alpha^l$
    if $0 < \gcd(\beta - 1, f)$ **and** $\gcd(\beta - 1, f) < n$ **then**
      Return $\gcd(\beta - 1, f)$
    end
  end
end

For example, let $f = (X - 1)(X + 1) = X^2 - 1$ and $p = 7$. Clearly 1 and -1 have different orders

(mod p). Let $l = 2$ then $\nu_2(p-1) = 1$. Let $d = (p-1)/2$ we find that $\alpha = X^d \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$. Then we can get a factor of f by computing $\gcd(f, X^d - 1) = \gcd(f, X^3 - 1 \pmod f) = \gcd(X^2 - 1, X - 1) = \gcd(X^2 - 1, X - 1) = X - 1$.

Theorem 2.32. *Let $\mathbb{F}_{p^k}$ be a finite field, so that the prime factorization of $p^k - 1$ is given, l a prime dividing $p^k - 1$, and $f \in L_n$ with two roots r_1, r_2 of f so that $\operatorname{ord}(r_1) \neq \operatorname{ord}(r_2)$ then Algorithm 4 factors f deterministically in $O(\log(p)n^2k^3)$ operations.*

Proof. If a, b are elements of $\mathbb{F}^*_{p^k}$ with distinct order then by unique factorization there exists a prime l and a positive integer i so that $l^i \mid \operatorname{ord}(a)$ and $l^i \nmid \operatorname{ord}(b)$ or $l^i \nmid \operatorname{ord}(a)$ and $l^i \mid \operatorname{ord}(b)$. Without loss of generality let l be the least prime factor of $p^k - 1$ so that $i = \nu_l(\operatorname{ord}(r_1))$ and $i > \nu_l(\operatorname{ord}(r_2))$ for some positive integer i and roots r_1, r_2 of f. Let $s = \nu_l(p^k - 1)$ and $d = (p^k - 1)/l^s$ then by Corollary 2.10

$$l^i \mid \operatorname{ord}\big(r_1^d\big) \text{ and } l^i \nmid \operatorname{ord}\big(r_2^d\big).$$

The components of $\alpha = X^d$ in $R = \mathbb{F}_{p^k}[X]/(f)$ are l^sth roots of unity. The component corresponding to r_1 is a primitive l^ith root of unity. The component corresponding to r_2 is not a primitive l^ith root of unity. Then

$$r_1^{l^{i-1}} \neq 1$$

$$r_2^{l^{i-1}} = 1.$$

In Algorithm 4 we repeatedly raise α to the power l and let $\beta = \alpha^l$. After doing this $i - 1$ times the component corresponding to r_1 is not congruent to 1 while the component corresponding to r_2 is congruent to 1 and so $\gcd(\beta - 1, f)$ is a nontrivial factor of f.

We now consider the number of operations used by the algorithm. We can compute $X^d \pmod f$ using a repeated squaring algorithm in $O(\log(d)(nk)^2) = O(\log(p)(nk)^2)$ operations by Theorem 2.29. Again using repeated squaring we can compute $\alpha^l, \alpha^{l^2}, \ldots, \alpha^{l^s}$ by first computing $\alpha^l \pmod{\langle f, p\rangle}$ and then finding $\alpha^{l^2}, \ldots, \alpha^{l^s}$ by repeatedly multiplying by α^l. These two steps take

$$\begin{aligned} &O(\log(l)(nk)^2 + s(nk)^2) = O(\log(l)(nk)^2 + \nu_l(p^k - 1)(nk)^2) \\ =&O(\log(p^k)(nk)^2 + \nu_l(p^k - 1)(nk)^2) = O(\log(p)n^2k^3) \end{aligned}$$

$\mathbb{F}_p$-operations. The gcd of two degree n polynomials over $\mathbb{F}_{p^k}$ takes $O((nk)^2)$ operations by Theorem 2.28 and there are s gcd computations within the while loop. Together the gcds take $O(s(nk)^2) = O(\log(p)n^2k^3)$ operations. □

Later in this chapter, in Algorithm 6 we give a more general way to factor polynomials based on the index of the roots. However, this type of factorization generally requires an mth nonresidue for some m dividing $p-1$. Such nonresidues cannot be constructed without ERH.

In the algorithms for factoring polynomials over finite fields in this thesis we do not need to directly compute the $\text{ind}(x)$ or $\text{ind}(f)$ for x an element of $\mathbb{F}_q$ or f an element of $\mathbb{F}_q[X]$. This is fortunate as there are no polynomial time algorithms to compute $\text{ind}(x)$. We give Algorithm 5 which computes the index of a finite field element modulo 2^s. We note that if x is an element of the finite field $\mathbb{F}_q$, $s = \nu_2(q-1)$, $d = 2^s$, $e = d^{-1} \pmod{(p-1)/2^s}$ then

$$\text{ind}(x) \equiv de \ \ \text{ind}(x) \equiv \text{ind}(x^{de}) \pmod{2^s}.$$

This congruence is used by the following algorithm.

Algorithm 5: TWO_IND

Input : $\mathbb{F}_q, \zeta, x$
  $\mathbb{F}_q$ a finite field
  ζ a quadratic nonresidue
  $x \in \mathbb{F}_q$
Output : $\text{ind}(x) \pmod{2^s}$ computed with respect to ζ with s the largest power of 2 dividing $q-1$

$s = \nu_2(q-1)$
$d = 2^s$
$e = d^{-1} \pmod{(p-1)/2^s}$
$x = x^{de}$
$t = 0$
for i **from** 1 **to** $s-1$ **do**
  $a = x^{(q-1)/2^i}$
  $b = \zeta^{t\cdot(q-1)/2^i}$
  if $ab^{-1} \neq 1$ **then**
    $t = t + 2^{i-1}$
  end
end
Return t

We give an example to illustrate Algorithm 5. Let $x = 1$ and let p be prime so that $\nu_2(p-1) = 3$ and ζ a quadratic nonresidue modulo p. Then $\zeta^{2^s} = x$ so we expect $\text{ind}(x) \pmod{2^s}$ to be 0. In the first iteration of the for loop in Algorithm 5 $a = 1, b = \zeta^{(p-1)/2} = -1$ (as ζ is a quadratic nonresidue) and t is not incremented. In the next iteration $a = 1, b = \zeta^{(p-1)/4}$ is a primitive fourth root of unity. Again t is not changed. In the last iteration of the loop $a = 1, b = \zeta^{(p-1)/8}$ a primitive 8th root of unity. The t returned by the algorithm is zero.

The problem of computing $\text{ind}(x)$ has been extensively studied due to its relevance in cryptography. Theorem 2.33 is given in [27]. We note that the number of operations depends on l. This is not an issue

when l is small but means that applying the theorem for each prime dividing $q-1$ is expensive if $S(q-1)$ is large.

Theorem 2.33. *Let $\mathbb{F}_q$ be a finite field, g be a primitive root in $\mathbb{F}_q^*$, l be a prime dividing p^k-1, and $\nu_l(p^k-1)$. Then there exists an algorithm, of which Algorithm 5 is a special case that computes* $\text{ind}(x) \pmod{l^s}$ *with respect to g in $O(lk^3 \cdot \nu_l(p^k-1)\log(p)) = O(lk^4 \cdot \log(p)^2)$ operations.*

Given a finite field $\mathbb{F}_{p^k}$ with $p^k - 1 = l_1^{s_1} \dots l_j^{s_j}$ we can apply Theorem 2.33 for each prime l dividing p^k-1. This gives

$$\text{ind}(x) \pmod{l_1^{s_1}}, \dots, \text{ind}(x) \pmod{l_j^{s_j}}.$$

We can then find $\text{ind}(x)$ using the Chinese remainder theorem. The number of operations required depends on the largest prime factor of $p-1$ which we denote $S(p-1)$. The total cost of computing $\text{ind}(x)$ using this algorithm is $O(S(p-1)k^4 \cdot \log(p)^2)$. The algorithm can be modified so that its running time is improved by a factor of $S(p-1)^{1/2}$ using the *baby step-giant step* method. This modification is the best *deterministic* algorithm for computing $\text{ind}(x)$.

The problem of finding roots of the polynomial $X^n - a$ for $a \in \mathbb{F}_{p^k}$ is closely related to computation of $\text{ind}(a) \pmod{l_1^{s_1}}, \dots, \text{ind}(a) \pmod{l_j^{s_j}}$. The following theorem generalizes the result [37] on computing square roots modulo a prime p.

Theorem 2.34. *Let p^k be a prime power, l be a prime such that l divides p^k-1. Given an lth nonresidue in $\mathbb{F}_{p^k}$ there is a deterministic algorithm to compute all solutions to $X^l - a$ in $O(lk^3 \cdot \nu_l(p^k-1)\log(p)) = O(lk^4 \cdot \log(p)^2)$ operations.*

We note that the number of operations is essentially a result of doing of doing $l \cdot \nu_l(p^k-1)$ exponentiations over $\mathbb{F}_{p^k}$. We give a proof of Theorem 2.34 for the case $l = 2$ over a prime field.

Proof. Let p be a prime. Suppose we need to find a root of the polynomial $f = X^2 - a$ for $a \in \mathbb{F}_p$. As $a^{(p-1)/2} \equiv 1 \pmod{p}$ if and only if a has a square root we may as well assume that a root of f exists. Equivalently, we can say that a is a quadratic residue as $\text{ind}(a) \equiv 0 \pmod{2}$. Let $s = \nu_l(p-1)$, $v \equiv 2^{-s} \pmod{(p-1)/2^s}$, $y = (p-1)/2^s$, and $z = y^{-1} \pmod{2^s}$. Then a can be written as a product of two elements α, ω in $\mathbb{F}_p$ such that α has odd order and ω has order not divisible by $(p-1)/2^s$. Let

$$\alpha = a^{2^s v}$$

$$\omega = a^{yz}$$

then $a = \alpha\omega$. Now we can construct a square root of a by constructing a square root of α and ω separately. Let $j = 2^{-1} \pmod{(p-1)/2^s}$ then α^j is a square root of α.

Suppose we have a quadratic nonresidue ζ then using Algorithm 5 we can find a positive integer k such that $\zeta^k = \omega$. We have $\text{ind}(a) = \text{ind}(\alpha) + \text{ind}(\omega)$. As a is a quadratic residue $\text{ind}(a) \equiv 0 \pmod 2$. As α has odd order $\text{ind}(\alpha) \equiv 0 \pmod 2^s$ so in particular $\text{ind}(\alpha) \equiv 0 \pmod 2$. Hence $\text{ind}(\alpha) \equiv 0 \pmod 2$ and $\zeta^{k/2}$ is a square root of α.

Multiplying the square roots of α and ω together we get that $b = \alpha^j \omega^{k/2}$ is a square root of a. The other set of all solutions to $X^2 - a$ is $\{b, -b\}$.

□

We note that although Theorem 2.34 is given for finding roots of $X^l - a$ for prime l it can be extended $X^n - a$ for arbitrary n. The idea is to factor n as $n = l_1^{e_1} \cdots l_k^{e_k}$. Then for prime l and positive integer k define $H_{l,k} = \{z \in \mathbb{F}_p, h \in H_{l,k-1} : z^l - h\}$ and $H_{l,0} = \{z \in \mathbb{F}_p : z^l - a\}$. We can recursively compute each of the sets $H_{l,k}$ using Theorem 2.34. Then the set of all solutions to $X^n - a$ is the direct product of $H_{l_1,e_1}, \ldots, H_{l_1,e_1}$.

The next algorithm factors polynomials over a finite field if $X^{(q-1)/l^s} \notin \mathbb{F}_q \pmod f$ where l^s is the largest power of a prime l dividing $q-1$. In Algorithm 6 MROOT(x, ζ, r) with x an element of $\mathbb{F}_q^*$, ζ an lth nonresidue, and r a divisor of $q-1$ is any algorithm that returns an rth root of x if it exists.

Algorithm 6: IND_FAC

Input: $\mathbb{F}_q, l, \zeta, f$
  $\mathbb{F}_q$ a finite field
  l a prime dividing $q-1$
  ζ an lth nonresidue in $\mathbb{F}_q$
  f a squarefree polynomial in L_n over $\mathbb{F}_q$
Output: A factor of f if a condition is met, otherwise FAIL

$s = \nu_l(q-1)$
$h = X^{(q-1)/l^s} \pmod{\langle f, \mathbb{F}_q \rangle}$
if $h \in \mathbb{F}_q$ **then**
  return FAIL
end
while $h \notin \mathbb{F}_q$ **do**
  /* The polynomial t is an rth root of h */
  $\beta = h$
  $h = h^l$
end
/* Construct ω an lth primitive root of unity from ζ */
$\omega = \zeta^{(q-1)/l}$
$z = \text{MROOT}(\text{h}, \zeta, \text{l})$
for $i = 0$ **to** $l-1$ **do**
  $g = \gcd(\beta - \omega^i z, f)$
  Return g
end

We give an example to illustrate this algorithm. Let p be a prime so that $\nu_2(p-1) = 4$ and z be a

quadratic nonresidue modulo p. Let a and b be distinct elements of $\mathbb{F}_p$ of odd order, and let

$$f = (X - az)(X - bz)\big(X - az^3\big)\big(X - bz^3\big)\big(X - az^{13}\big)\big(X - bz^{13}\big).$$

We compute the index of the roots with respect to z:

$$\operatorname{ind}(az^i) \equiv \operatorname{ind}(a) + \operatorname{ind}(z^i) \equiv \operatorname{ind}(z^i) \pmod{2^s}$$

where the last equality holds because a has odd order. Similarly we have $\operatorname{ind}(bz^i) \equiv \operatorname{ind}(z^i) \pmod{2^2}$. Hence $\operatorname{ind}(f) \equiv (1, 1, 3, 3, 13, 13) \pmod{2^s}$ where ind is computed with respect to the quadratic nonresidue z. In base two these indexes are $I = [i_1, i_2, i_3, i_4, i_5, i_6] = [1, 1, 11, 11, 1101, 1101]$. In this representation the lowest bit of each element of I is 1. Consider what happens when we apply Algorithm 6 to this polynomial. As all the indexes are equal on this bit, in Algorithm 6 the element h^{2^3} is in $\mathbb{F}_p$. So the while loop in the algorithm terminates after at most 3 iterations.

As the second lowest bit is different for i_1 and i_3 the element h^{2^2} is not in $\mathbb{F}_p$ so the while loop is terminated after exactly three iterations. The algorithm recovers the factors h_1, h_2 whose roots have the second lowest bit in their base 2 indexes equal to 0 and 1 respectively. These are

$$h_1 = \big(X - az^3\big)\big(X - bz^3\big)\big(X - az^{13}\big)\big(X - bz^{13}\big) \qquad h_2 = (X - az)(X - bz).$$

The indexes of h_1 in base two are $[11, 11, 1101, 1101]$ so the distinct indexes differ on the second lowest bit. Hence when Algorithm 6 is applied to h_1 the while loop is terminated after two iterations. We get the factorization $h_1 = g_1 g_2$ where

$$g_1 = \big(X - az^3\big)\big(X - bz^3\big) \qquad g_2 = \big(X - az^{13}\big)\big(X - bz^{13}\big).$$

The factorization $f = g_1 g_2 h_2$ is as far as we can go using Algorithm 6 with $l = 2$; let a and b be arbitrary roots of distinct elements of the set $\{g_1, g_2, h_2\}$ then $\operatorname{ind}(a) \not\equiv \operatorname{ind}(b) \pmod{2^s}$.

Theorem 2.35. *Let $\mathbb{F}_{p^k}$ be a finite field, f a squarefree polynomial in L_n, l be a prime such that $p^k - 1 = ml^s$ where l does not divide m, ζ an lth nonresidue, and $X^m \neq 1 \pmod{f}$. Then Algorithm 6 factors f deterministically in $O(k^4 \log(p)^2(n^2 + l))$ operations.*

Proof. Let $\alpha = X^{(q-1)/l^s} \pmod{f}$ and $v = M(\alpha)$, r be a root of v, and $[\operatorname{ind}(r)[l^0], \ldots, \operatorname{ind}(r)[l^{s-1}]] =$

$[a_0, \ldots, a_{l-1}]$. Then by Lemma 2.16

$$\begin{aligned}
[\mathrm{ind}(r)[l^0], \ldots \quad , \mathrm{ind}(r)[l^{s-1}]] \quad &= [a_0, a_1 \ldots, a_{s-1}] \\
[\mathrm{ind}(r^l)[l^0], \ldots \quad , \mathrm{ind}(r^l)[l^{s-1}]] \quad &= [0,\ a_0, \ldots, a_{s-2}] \\
\quad \vdots \qquad\qquad\qquad\qquad\qquad\quad &\quad \vdots \\
[\mathrm{ind}(r^{l^{s-1}})[l^0], \ldots, \mathrm{ind}(r^{l^{s-1}})[l^{s-1}]] &= [0,\ 0, \ldots,\ a_0].
\end{aligned}$$

There is some least positive integer i so that for any pair of roots r_1, r_2 we have

$$[\mathrm{ind}(r_1^{l^i})[l^0], \ldots, \mathrm{ind}(r_1^{l^i})[l^{s-1}] = [\mathrm{ind}(r_2^{l^i})[l^0], \ldots, \mathrm{ind}(r_2^{l^i})[l^{s-1}].$$

Let $\beta = \alpha^{l^{i-1}}$, and $z \in \mathbb{F}_p$ be an lth root of β^l. Let b be an arbitrary component of β then by Lemma 2.17 for every $0 < k < s-2$ we have $\mathrm{ind}(b)[l^k] = \mathrm{ind}(z)[l^k]$. Let ω be an lth primitive root of unity and i a positive integer then $\mathrm{ind}(\omega^i z)[l^{s-1}] = i \cdot \mathrm{ind}(\omega)[l^{s-1}] + \mathrm{ind}(z)[l^{s-1}]$. We choose i so that $\mathrm{ind}(b)[l^{s-1}] = \mathrm{ind}(\omega^i z)[l^{s-1}]$. Hence as b and z are both l^s roots of unity $\mathrm{ind}(b) = \mathrm{ind}(\omega^i z)$ and $b = \omega^i z$. So for this i we have $\beta - \omega^i z$ is 0 on one component. However, it cannot be zero on every component or β would be in $\mathbb{F}_p$. Therefore, $\gcd(\beta - \omega^i z, f)$ is a factor of f.

We now consider the number of operations. h can be computed in $\log(p)(n^2k^3)$ by Theorem 2.29 and $s = \nu_l(p^k - 1) = O(k \log(p))$ so all the powers of h can be computed in $\log(p)^2(n^2k^4)$ operations. MROOT uses $O(l \cdot \log(p)^2 k^4)$ operations by Theorem 2.34. Computing each of the polynomials g requires a gcd operation which takes $O((nk)^2)$ operations by Theorem 2.28 so $O(l(nk)^2)$ operations are needed to compute them all. Hence Algorithm 6 uses $O(\log(p)^2 k^4 (n^2 + l))$ operations. □

In the algorithms in Chapter 3 and Chapter 4 we are often given a squarefree polynomial f that is a product of linear factors over a finite field to factor. We construct a polynomial g so that $g = (X - r_1)^{e_1} \cdots (X - r_n)^{e_n}$ where $r_1, \ldots, r_n$ are the roots of f. It is sometimes possible to factor g which in turn allows a factorization of f.

Definition 2.36. Given a squarefree polynomial $f \in L_n$ over a finite field $\mathbb{F}_q$ a polynomial g is *flat* if $f = \gcd(f, g)$ and every root of g has the same multiplicity.

Algorithm 7: NON_FLAT

Input: $\mathbb{F}^*_{p^k}, f, g$
$\mathbb{F}^*_{p^k}$ a finite field
$f = \prod_{i=1}^{k} f_i$ with f_i irreducible polynomials
$g = \prod_{i=1}^{k} f_i^{j_i}$
Output : 1 if f is not flat, otherwise a nontrivial factor of f

$h = g$
while $f \mid \gcd(h, f)$ **do**
  $h = h/f$
end
Return $\gcd(h, f)$

For example let $f = (X - r_1)(X - r_2)$ and $g = (X - r_1)(X - r_2)^2$ over some prime field $\mathbb{F}_p$. As f is not flat we can factor it using Algorithm 7. The polynomial h is initialized to g and f divides h so $h = h/f = (X - r_2)$. Then f does not divide this new h so $(X - r_2)$ is returned.

Lemma 2.37. *Let $\mathbb{F}_q$ be a finite field and $f, g \in L_n$ over $\mathbb{F}_q$ so that g is not flat with respect to f. Then Algorithm 7 returns a factor of f in $O(m(n+m)^2k^2)$ operations.*

Proof. Let $g = \gcd(g, f) = {a_1}^{e_1} \cdots {a_k}^{e_k}$ with each a_i a factor of f of degree one and $e_1 \leq e_2 \leq \cdots \leq e_k$. Then f divides $\gcd(g, f)$. In the algorithm h is initialized as $g = {a_1}^{e_1} \cdots {a_k}^{e_k}$. Now we consider the factorization of h as $h = {a_1}^{e_1} \cdots {a_k}^{e_k}$ after each iteration of the while loop. Each division of h by f in the while loop reduces each of the e_i by one. This is repeated until $e_1 = 0$. We need to show that if g is not flat with respect to f at this point $\gcd(h, f)$ is a nontrivial factor of f. Note that $\gcd(h, f)$ is a squarefree divisor of f. As $e_1 = 0$ we have f does not divide $\gcd(h, f)$. As g is not flat there are at least two distinct elements in the set $\{e_1, \ldots, e_k\}$ so $\gcd(h, f)$ is not equal to f. Hence $\gcd(h, f)$ is a nontrivial factor of f.

We now consider the number of operations. Let $m = \deg(g)$. As the e_i are reduced by one after each iteration the loop iterates at most m times in total. Each iteration uses one division and one gcd both of which take $O((n+m)^2k^2)$ operations. □

2.3 Graph Theory Background

In this section we give some definitions and lemmas related to graph theory. A graph is a pair (V, E) where V is a nonempty set and E is a set of unordered pairs of elements of V. In general the vertices can be arbitrary, but in this thesis they will always be indeterminants or positive integers less than p for some given prime p. These are the *vertices* and *edges* of a graph, respectively. Two vertices a and b are called *adjacent* if $(a, b) \in E$. If a pair of vertices is adjacent then the pair are referred to as neighbors. A *cycle* of a graph G is a sequence of vertices $v_1, \ldots, v_n$ such that no pair of distinct vertices are equal

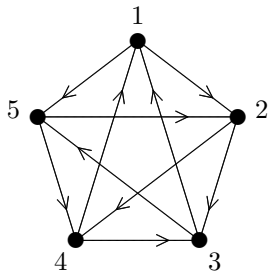

Figure 1: A tournament on 5 vertices.

except the first and the last. An *edge cycle* on a graph G is a sequence of edges such that for any vertex v in G, v is in two of the edges of the cycle.

The number of edges adjacent to a vertex v is the *degree* of v. A graph is *regular* if every vertex has the same degree. A directed graph (digraph) is a pair (V, E) so that V is nonempty and E is a set of ordered pairs of elements of V. The sets V and E are the vertices and the edges (also called arcs) of G. The indegree of a vertex v is the number of elements of E that have v as their second element, and the outdegree is the number of elements of E that have v as their first element. A digraph is *regular* if the indegree and outdegree of every vertex are equal to a positive integer k. A *circuit* of a digraph G is a sequence of vertices $v_1, \ldots, v_n$.

This rest of this section discusses a type of digraph called tournaments. These graphs are only referenced to in the chapter on results on FPFF that are conditional on ERH. A *tournament* is a directed graph in which every pair of vertices is joined by exactly one arc. A vertex a *dominates* a vertex b if there is an arc from a to b. The *score* $s(v)$ of a vertex v is the number of vertices dominated by it. A tournament on n vertices is called an n-tournament. The *converse* T' of a tournament T has the same vertex set as T but all the arcs are reversed. Figure 1 provides an example of a 5-tournament

The number of tournaments is $2^{\binom{n}{2}}$. The number of regular tournaments is determined in [23]. See [28] for a survey of results on tournaments.

Let G be a graph or digraph with vertices $\{Y_1, \ldots, Y_n\}$ for some positive integer n. If $\mathbb{F}_p$ is a prime field containing all the Y_i then the *vertex polynomial* is $\prod_{i=1}^{n}(X - Y_i)$. From now on whenever we give an algebraic expression involving a set of vertices of some graph $Y_1, \ldots, Y_n$ it is implicit that Y_i are elements of a prime field.

Definition 2.38. Let G be an undirected graph with vertices $\{Y_1, \ldots, Y_n\}$ and $*$ be a commutative

binary operation. The *edge polynomial* of G with respect to $*$ is

$$\prod_{(Y_i,Y_j)\in E} (X - Y_i * Y_j).$$

For example, the edge polynomial of the complete undirected graph on n vertices is

$$\prod_{1\leq i<j<n} (X - Y_i * Y_j).$$

In particular, the arc polynomial corresponding to the complete undirected graph on 4 vertices with respect to the operation of ordinary addition is $(Y - (r_1 + r_2))(Y - (r_1 + r_3))(Y - (r_1 + r_4))(Y - (r_2 + r_3))(Y - (r_2 + r_4))(Y - (r_3 + r_4))$.

Any graph on n vertices can be given as a list where the ith entry is all the edges which have the ith vertex as an endpoint. The edge polynomial may be thought of as an encoding of the adjacency list representation of a graph. We now define an analogous polynomial for digraphs.

Definition 2.39. Let G be a digraph with vertices $\{Y_1, \ldots, Y_n\}$ and $\circ$ be a noncommutative binary operation. The *arc polynomial* of G with respect to $*$ is

$$\prod_{(Y_i,Y_j)\in E} (X - Y_i * Y_j).$$

Let G be a tournament with vertices $\{Y_1, \ldots, Y_n\}$ for some positive integer n. The *score polynomial* of a tournament is $\prod_{i=1}^{n} (X - Y_i)^{s(Y_i)}$.

3 Related Work on FPFF

3.1 Deterministic Algorithms

In this section we summarize the results in the literature on FPFF that are probabilistic, deterministic, and deterministic under certain conjectures.

Probabilistic factorization algorithms were given in [41] and [6] that use a number of operations polynomial in $\log(q)$ and $\deg(f)$. A more efficient algorithm was given by [9]. These probabilistic algorithms are of Las Vegas type, if a solution is produced then it is correct. The asymptotically fastest algorithm for factoring polynomials is given in [19].

In the unconditional case there has not been much progress on FPFF. No unconditional and deterministic algorithm for factoring polynomials over $\mathbb{F}_p$ is known that runs in subexponential time in either the degree of the polynomial or $\log(p)$. The oldest general result is a result of Berlekamp [5],[6].

Theorem 3.1. *Let $\mathbb{F}_q[X]$ be a finite field, and f be a polynomial of degree n over $\mathbb{F}_q$. There exists an algorithm that factors f deterministically in $O((n+q)n^2)$ operations.*

We note that in the literature the term "Berlekamp's algorithm" is used to refer to both the probabilistic and deterministic algorithm for FPFF given in [6]. The following theorem is given in [34].

Theorem 3.2. *Let $\mathbb{F}_{p^k}$ be a finite field and f a degree n polynomial in $\mathbb{F}_{p^k}[X]$. Then f can be factored deterministically in*

$$\log(p)\tilde{O}((nk)^2) + p^{1/2}\log(p)\tilde{O}((nk)^{3/2})$$

operations.

An algorithm for factoring almost all polynomials deterministically is given in [33] and this result is extended in [20].

Many results on deterministic FPFF pertain to finding roots of polynomials in $\mathbb{F}_q[X]$ of the form $X^n - z$ with z a finite field element. In this case a root of the polynomial is called an nth root of z. Deterministic algorithms for computing nth roots exist which run in polynomial time in special cases. In [32] an algorithm for computing a square root of x (if one exists) in $\mathbb{F}_p$ that uses $O((x^{1/2+\epsilon}\log(p))^9)$ operations is given as an application of an algorithm for counting points on elliptic curves. Using related ideas, [25] shows that for any odd prime l and prime p such that $p \equiv 1 \pmod{l}$ an lth primitive root of unity in $\mathbb{F}_p$ can be computed in $\text{poly}(\log(p), l)$. A deterministic algorithm to compute square roots in finite fields that runs in polynomial time for some special finite fields is given in [36].

Many of the results on factoring polynomials over finite fields depend on the Riemann Hypothesis (RH) or generalizations of it. Before providing these results we give the statement of RH and two of the

commonly used generalizations.

3.2 Riemann Hypothesis

Most of the results on factoring polynomials over finite fields assume either the Riemann Hypothesis or a generalization of it. The two generalizations that are commonly used are the Extended Riemann Hypothesis (ERH) and the Generalized Riemann Hypothesis (GRH). We give the statement of these conjectures in their simplest form. The definitions and theorems in this section follow Chapter 8 of [4]. Let

$$li(x) = \int_2^x \frac{dt}{\log(t)}.$$

and let $\pi(x)$ be the number of primes less than or equal to x. First we state the ordinary Riemann Hypothesis.

Riemann Hypothesis: Let x be a positive integer. For any $\epsilon > 0$

$$\pi(x) = li(x) + O(x^{\frac{1}{2}+\epsilon}).$$

It was proved by Dirichlet that if $\gcd(a, n) = 1$ then there are infinitely many primes congruent to a modulo n. (These are called primes in an arithmetic progression). Let $\pi(x, n, a)$ be the number of primes p less than or equal to x such that $p \equiv a \pmod{n}$. The Extended Riemann Hypothesis (ERH) is a generalization of the Riemann hypothesis which can be stated in terms of primes in arithmetic progressions. Recall that $\phi(n)$ is the Euler totient function. Extended Riemann Hypothesis: Let x, n, a be positive integers and a and n be relatively prime, then for any $\epsilon > 0$

$$\pi(x, n, a) = \frac{li(x)}{\phi(n)} + O(x^{\frac{1}{2}+\epsilon}).$$

Any finite extension of the rationals of finite degree is called an *algebraic number field*. The generalization of the Riemann Hypothesis to algebraic number fields is called the Generalized Riemann Hypothesis.

Algebraic number fields are often represented in the form $\mathbb{Q}[X]/(f)$ where f is an irreducible polynomial in $\mathbb{Z}[X]$. The *degree* of an algebraic number field is its dimension as a vector space over $\mathbb{Q}$. If k is an algebraic number field of degree n there are n field homomorphisms from k into $\mathbb{C}$. These are given by the factorization of f and are called *embeddings*.

Definition 3.3. Let k be an algebraic number field of degree n and $\alpha_1, \ldots, \alpha_n$ be its embeddings. The

norm of $x \in k$ is defined

$$N(x) = \prod_{i=1}^{n} \alpha_i(x).$$

Theorem 3.4. *If k is an algebraic number field of degree n and $x \in k$, then there is a unique monic polynomial $f \in \mathbb{Q}[X]$ of degree less than or equal to n such that $f(x) = 0$ and for any polynomial $g \in \mathbb{Q}[X]$ if $g(x) = 0$ then $f \mid g$. Then f is called the* minimal polynomial *of x.*

Let k be an algebraic number field and R be the subset of k such that the minimal polynomial of every element in R is in $\mathbb{Z}$. Then R forms a subring and is called the ring of *algebraic integers* of k.

A nonzero ideal A of the ring of integers R of an algebraic number field is a *prime ideal* if A/P is an integral domain. An analogue of the prime number theorem is known for prime ideals. For any algebraic number field the number of prime ideals is infinite.

For an algebraic number field k let $\pi_k(x)$ be the number of prime ideals with norm less than or equal to x.

Generalized Riemann Hypothesis: Let k be an algebraic number field, then for any $\epsilon > 0$

$$\pi_k(x) = li(x) + O(x^{\frac{1}{2}+\epsilon})$$

3.3 Results Depending on RH, ERH, GRH

Many algorithms for polynomial factorization over $\mathbb{F}_p$ require the construction of nonresidues. For example, the first step of the algorithm of Theorem 2.34 for computing square roots in $\mathbb{F}_p$ (i.e. finding a root of the polynomial $X^2 - a$ for $a \in \mathbb{F}_p$) is finding a quadratic nonresidue. It is easy to construct a nonresidue probabilistically. Finding nonresidues deterministically is more challenging. In prime fields testing $1, 2, \ldots$ until an mth nonresidue is found is the best known approach known. Typically, one way to bound the size of the set required is by bounding a character sum. We define the *Jacobi symbol* as χ. We let $\chi(x) = 1$ if x is a quadratic residue modulo p and $\chi(x) = -1$ if x is a quadratic nonresidue modulo p. Suppose we can find M so that

$$\sum_{x=1}^{m} \chi(x) < M$$

then this forces the existence of a quadratic nonresidue modulo p less than M.

For many character sums that arise in polynomial factorization algorithms there is a significant gulf between what can be proved unconditionally, and what can be proved on conditions such as the Riemann Hypothesis, or at least motivated heuristically. For example, the best bound on the sum above is $O\left(p^{1/4\sqrt{e}+\epsilon}\right)$ for any $\epsilon > 0$ [8]. Under ERH the sum is bounded by $O(\log(p)^2)$.

Theorem 3.5. *Let m be a divisor of $p-1$. The least mth nonresidue modulo p is $O(\log(p)^2)$ under ERH.*

Theorem 3.5 was given in [1] in the case $m = 2$. A proof of the full theorem is given in [4] Chapter 8.

Theorem 3.6. *The following have algorithms that use a polynomial number of operations in q and $\log(p)$ under GRH.*

- *Given primes q, p factor $\Phi_q \pmod p$*
- *Given primes q, p construct a qth nonresidue in $\mathbb{F}_p[X]/(h)$ where h is an irreducible factor of $\Phi_q \pmod p$.*

Using this algorithm mth nonresidues in $\mathbb{F}_q[X]$ can be constructed in $\text{poly}(m, \log(q))$ field operations.

The problems of finding lth nonresidues and taking lth roots are closely related. Given an lth nonresidue, lth roots can found deterministically. In the other direction, if a primitive lth root of unity ω_l in $\mathbb{F}_q$ is known, an algorithm for taking lth roots could be used to repeatedly take roots of ω_l until an lth nonresidue is found. In the case $l = 2$, -1 is a primitive 2th root of unity so the problem of computing quadratic nonresidue reduces to the problem of computing square roots.

The next theorem was proved in [29]. We will also prove this result in Chapter 5. Recall that we defined $L_n = \left\{f : f = \prod_{i=1}^{n}(X - a_i), a_i \in \mathbb{F}_q^*\right\}$.

Theorem 3.7. *Let f be a squarefree polynomial in L_n over $\mathbb{F}_p$. Let r be a prime divisor of n and suppose that the rth cyclotomic field F over $\mathbb{F}_p$ and an rth nonresidue in F are given. Then we can find a nontrivial factor of f in $\text{poly}(\log(p), n^r)$ time under GRH.*

The best result on factoring general polynomials under GRH is the following result, given in [12].

Theorem 3.8. *Let $\mathbb{F}_{p^k}$ be a finite field and f be a degree n polynomial over $\mathbb{F}_{p^k}$. Then f can be factored deterministically in $\text{poly}(\log(p), n^{\log(n)})$ operations under GRH.*

The following theorem is given in [13]. It gives a condition for factorization on the Galois group.

Theorem 3.9. *Let f be a squarefree polynomial in L_n over $\mathbb{F}_p$ with solvable Galois group over $\mathbb{Q}$. Then f can be completely factored into irreducible factors in $\text{poly}(\log(p), n)$ operations under GRH.*

In [15] an algorithm for factoring a class of polynomials which are called *super-square balanced* in polynomial time under GRH is given. A conjecture is given under which the algorithm would factor any polynomial in deterministic polynomial time under GRH. In [31] this approach is generalized and in [10] an algorithm related to [15] is given. These approaches start from the following idea. Given a polynomial $f = \prod_{i=1}^{n}(X - r_i)$ then we can define a corresponding directed graph G with vertices labeled $\{r_1, \ldots, r_n\}$

such that there is an arc $r_i r_j$ for $i \neq j$ if $\chi(r_i - r_j) = 1$ and an arc $r_j r_i$ otherwise. The score $s(v)$ of a vertex v is the number of arcs leaving it. Let $V(G)$ be the set of vertices of of G. It is possible to construct the polynomial

$$h = \prod_{v \in V(G)}^{n} (X - v)^{s(v)}.$$

Recall that we call a polynomial flat if the multiplicity of every factor is the same. If there is a pair of vertices with different scores then h is not flat and so f can be factored by Algorithm 7. The papers [15] and [10] develop different approaches for dealing with the case that $s(v)$ is the same for every vertex. The next two theorems make use of an algebraic-combinatorial generalization of [10]. The following theorem is given in [18].

Theorem 3.10. *If $n > 2$ is prime, r the largest prime factor of $(n-1)$ and f is a degree n polynomial over $\mathbb{F}_p$ then a f can be factored deterministically in $\text{poly}(\log(p), n^r)$ operations under GRH.*

The next theorem is given in [2].

Theorem 3.11. *Let f be a polynomial of prime degree n over $\mathbb{F}_q$. Assume that $n-1$ has a r-smooth divisor s, with $s \geq \sqrt{n/l} + 1$ and $l \in \mathbb{N} > 0$. Under GRH we can find a nontrivial factor of f deterministically in $\text{poly}(\log(q), n^{r+\log(l)})$ operations.*

The following theorem was proved in [39]. It shows that for special finite fields any polynomial can be factored. The main tool used in the proof is Algorithm 6.

Theorem 3.12. *Every $f \in \mathbb{F}_p[X]$ can be factored deterministically in $\text{poly}(S(p-1), \log(p), n)$ operations under ERH.*

Proof. We can factor $p - 1 = l_1^{e_1} \cdots l_k^{e_k}$ in the time bound by trial division. By Theorem 2.31 we may as well assume that f is a squarefree polynomial in L_n over $\mathbb{F}_p$ and so every element in $\text{ind}(f) \equiv [\text{ind}(r_1), \ldots, \text{ind}(r_n)] \pmod{p-1}$ is distinct. We want to show that there exists l dividing $p-1$ so that at least two entries in $\text{ind}(f) \equiv [\text{ind}(r_1), \ldots, \text{ind}(r_n)] \pmod{l^s}$ are distinct. Suppose no such l exists, then in particular we have $\text{ind}(r_1) \equiv \text{ind}(r_2) \pmod{l^s}$ for every l dividing $p-1$. Hence by the Chinese remainder theorem we have $\text{ind}(r_1) \equiv \text{ind}(r_2) \pmod{p-1}$. However, since $\text{ind}(r_1) < p-1$ and $\text{ind}(r_2) < p-1$ this implies $r_1 = r_2$. As f is squarefree $r_1 = r_2$ is a contradiction and so such an l must exist.

Our goal now is to factor the polynomial f using Theorem 2.35 where l is the prime given as input to the algorithm. In order to apply Theorem 2.35 we need an lth nonresidue for each l dividing $p-1$. By Algorithm 6 under ERH we can construct each lth nonresidue in $O(\log(p)^2)$ operations and so we can construct the set of nonresidues in $O(\log(p)^3)$. Then we apply Algorithm 6 to get a factor of f. By Theorem 2.35 the factor is constructed in $\text{poly}(S(p-1), \log(p), n)$ operations. □

In Chapter 4.1 we remove the condition of ERH when $\gcd(n, \phi(p-1)) = 1$.

4 Factoring Polynomials Unconditionally

Section 4.1 gives background information on some operations that are useful for polynomial factorization. We use these composed operations in both Chapter 4 and Chapter 5.

Through the results of Chapter 2 we know that given a nontrivial factor of a polynomial f of degree n over a finite field $\mathbb{F}_{p^k}^*$, in order to construct a nontrivial factor of f it suffices to find a root of a polynomial h in the following set

$$L_m = \{f = \prod_{i=1}^{n} (X - a_i) : f \in \mathbb{F}_p[X], a_i \in \mathbb{F}_p\}$$

with $m \leq n$ and which is squarefree. Given a root of h we can work backwards to construct a factor of f.

In Section 4.3 we give the main contribution of this chapter Theorem 4.15. Earlier we defined $S(x)$ to be the largest prime factor of x. Recall Theorem 3.12 which showed that any polynomial in L_n over $\mathbb{F}_p$ can be factored deterministically in $\text{poly}(S(p-1), \log(p), n)$ operations under ERH. For polynomials with degree coprime to $\phi(p-1)$ Theorem 4.15 removes the requirement of ERH. In Section 4.4 we give a new result in Theorem 4.21 which regards polynomials of bounded degree. We show that bounded degree polynomials can be factored deterministically in $O\big(p^{1/4+o(1)}\big)$ operations.

4.1 Composed Operations

A natural starting point for trying to construct a deterministic polynomial time factorization algorithm is to extend a deterministic algorithm $\mathcal{A}$ that takes $f \in \mathbb{F}_p[X]$ as input and outputs a nontrivial factorization for some $f \in \mathbb{F}_p[X]$ when a nontrivial factorization of f does exist, by doing some algebraic manipulation of f to produce a new polynomial g, then applying $\mathcal{A}$ to g to get a factor h, and then doing another manipulation on h to produce a factor of f.

For example, $\gcd(f, X^{(p-1)/2} - 1)$ may produce a factor of f. If it does not, then let $f_a = f(X + a)$ for random a in $\mathbb{F}_p$. If the gcd gives a factor h of f_a then $h(X - a)$ is a factor of f. This was first observed by Legendre in [40]. A related deterministic factorization algorithm was given by Shoup in [33].

This idea can be be formalized in the following way. Let $(\mathcal{A}, \mathcal{S}, \mathcal{T})$ be 3 deterministic polynomial time algorithms so that $\mathcal{A}$ takes $\mathbb{F}_p$ polynomials as input and returns either nothing or a factor of the input polynomial, $\mathcal{S}$ and $\mathcal{T}$ take pairs of $\mathbb{F}_p$ polynomials as input and output $\mathbb{F}_p$ polynomials and if $s = \mathcal{A} \circ \mathcal{S}(f, h)$ is a nontrivial factor of $\mathcal{S}(f, h)$ then $\mathcal{T}(f, s)$ is a nontrivial factor of f.

We now introduce five operations on polynomials. These operations are called the *composed power*, *composed sum*, *composed difference*, *composed product*, and *composed quotient*, respectively. Let f and g be polynomials over $\mathbb{F}_q$ with factorizations $f = \prod_\alpha (X - \alpha)$ and $g = \prod_\beta (X - \beta)$ in an algebraic closure of $\mathbb{F}_q$. Then

$$f^{(k)} = \prod_{\alpha}(X - \alpha^k),$$

$$f \oplus g = \prod_{\alpha}\prod_{\beta}(X - (\alpha + \beta)), \qquad f \ominus g = \prod_{\alpha}\prod_{\beta}(X - (\alpha - \beta)),$$

$$f \otimes g = \prod_{\alpha}\prod_{\beta}(X - \alpha\beta), \qquad f \oslash g = \prod_{\alpha}\prod_{\beta}(X - \frac{\alpha}{\beta}).$$

For example, let $f = \Phi_8(X) = X^4 + 1 = (X - \omega)(X - \omega^3)(X - \omega^5)(X - \omega^7)$ where ω is an 8th root of unity. Then

$$f \oslash f = \big((X-1)\big(X-\omega^2\big)\big(X-\omega^4\big)\big(X-\omega^6\big)\big)^4 = (X-1)^4(X+1)^4(X^2+1)^4.$$

Computing the composed product efficiently was investigated in [7] and [14]. Composed products are connected to the minimal polynomials of sums and products of algebraic numbers [22]. Composed products are also useful for explicit constructions of factors of cyclotomic polynomials [38]. Other references for composed operations can be found in [24]. The theorems in this section are given in these papers.

The next two theorems deal with the case where the two polynomials in the composed operation are equal.

Theorem 4.1. *Let $f \in L_n$ such that there exists a polynomial $g \in \mathbb{F}_p[X]$ with $\deg(g) > 1$ and $g \mid f$. Let $\diamond$ denote one of the four operations $\oplus, \ominus, \otimes, \oslash$. Then $(g \diamond g) \mid (f \diamond f)$.*

Theorem 4.2. *Let $f = \prod_{i=1}^{n}(X - \alpha_i)$ be a degree n polynomial over an algebraic closure of $\mathbb{F}_q$ then $f \oplus f$, $f \ominus f$, $f \otimes f$,$f \oslash f$ have degree n^2 and can be written*

$$\begin{aligned}
f \oplus f &= \prod_{i=1}^{n}(X - 2\alpha_i)\prod_{i<j}(X - (\alpha_i + \alpha_j))^2 \\
X^{-n}(f \ominus f) &= \prod_{i<j}\big(X^2 - (\alpha_i - \alpha_j)^2\big) \\
f \otimes f &= \prod_{i=1}^{n}\big(X - {\alpha_i}^2\big)\prod_{i<j}(X - (\alpha_i\alpha_j))^2 \\
(X-1)^{-n}(f \oslash f) &= \prod_{i<j}\big(X^2 - X(\alpha_i + \alpha_j^{-1}) + 1\big)
\end{aligned}$$

We now prove two of these statements.

Proof. Let $A = \{\alpha_1, \ldots, \alpha_n\}$. We have $f \oplus f$ satisfies

$$f \oplus g = \prod_{\alpha \in A} \prod_{\beta \in A} (X - (\alpha + \beta)) = \prod_{\alpha \in A} (X - 2\alpha) \prod_{\alpha_1 \in A, \alpha_2 \in A, \alpha_1 \neq \alpha_2} (X - (\alpha_1 + \alpha_2)).$$

In the product $\prod_{\alpha_1 \neq \alpha_2} (X - (\alpha_1 + \alpha_2))$ each term $X - (\alpha_1 + \alpha_2)$ appears twice so

$$f \oplus f = \prod_{i=1}^{n} (X - 2\alpha) \prod_{i<j} (X - (\alpha_i + \alpha_j))^2.$$

We have $f \ominus g$ satisfies

$$f \ominus f = \prod_{\alpha} \prod_{\beta} (X - (\alpha - \beta)).$$

For each term $r_1 = X - (\alpha_1 - \alpha_2)$ there is a corresponding term $r_2 = X - (\alpha_2 - \alpha_1)$ and so $r_1 r_2 = X^2 - (\alpha_1 - \alpha_2)^2$ is a factor so

$$f \ominus f = X^n \prod_{i<j} (X^2 - (\alpha_i - \alpha_j)^2).$$

The other statements are proved in a similar way. □

Theorem 4.3. *Let f be a squarefree degree n polynomial with roots A in an algebraic closure of $\mathbb{F}_q$ then $f \oplus f$, $f \ominus f$, $f \otimes f$,$f \oslash f$ have degree n^2 and can be written*

$$\begin{aligned} f \oplus f &= \prod_{\alpha \in A} f(X - \alpha), \\ f \ominus f &= \prod_{\alpha \in A} f(X + \alpha), \\ f \otimes f &= f(0)^n \prod_{\alpha \in A} f(X\alpha^{-1}), \\ f \oslash f &= f(0)^{-n} \prod_{\alpha \in A} f(X\alpha). \end{aligned}$$

Proof. For the first equality we have

$$f \oplus f = \prod_{\alpha \in A} \prod_{\beta \in A} (X - (\alpha + \beta)) = \prod_{\alpha \in A} \prod_{\beta \in A} ((X - \alpha) - \beta) = \prod_{\alpha \in A} f(X - \alpha).$$

The other equalities are proved similarly. □

For example, if $f = (X - r_1)(X - r_2)$ then $f(0) = r_1 r_2$, $f \otimes f = (X - r_1^2)(X - r_2^2)(X - r_1 r_2)^2 =$

$r_1 r_2 (X/r_1 - r_1)(X/r_1 - r_2)(X/r_2 - r_2)(X/r_2 - r_1) = f(0) f(X/r_1) f(X/r_2)$

Let $\mathbb{F}_q$ be a finite field, f be a squarefree polynomial in L_n with roots $r_1, \ldots, r_n$, $R = \mathbb{F}_q[X]/(f)$, and $\alpha \in R$. The element α has a tuple of components $(\alpha_1, \ldots, \alpha_n)$ with $\alpha_i = \alpha(r_i)$. Recall that the minimal polynomial of α is defined

$$M_X(\alpha) = (X - \alpha_1) \cdots (X - \alpha_n).$$

Theorem 4.4. *Let p be a prime, k be a positive integer, and f be a squarefree polynomial in L_n, let $R = \mathbb{F}_{p^k}[X]/(f)$, and $\alpha \in R$. Then $M_Y(\alpha) = \mathrm{Res}(f, Y - \alpha)$ with the resultant computed with respect to X and $M_Y(\alpha)$ can be computed in $O(n^5 k^3)$ operations.*

Proof. Let $T(n)$ be the number of $\mathbb{F}_p$ operations to compute the product of two polynomials of degree n over $\mathbb{F}_{p^k}$. By the straightforward algorithm for multiplying polynomials we have $T(n) = O((nk)^2)$. Recall that the resultant is given by the determinant of the Sylvester matrix which is a square matrix of $n + m$ entries. In this case the entries of this matrix are polynomials of degree less than n. It is well known that the determinant of an $i \times i$ matrix with entries in a ring R can be computed in $O(i^3)$ ring operations, so $O((n+m)^3 T(n+m)) = O(n^5 k^3)$ $\mathbb{F}_p$-operations are required. □

Now we consider the issue of computing the composed operations.

Theorem 4.5. *Let f and g be polynomials in $\mathbb{F}_q[X]$ then*

$$\begin{aligned}
f \oplus g &= \mathrm{Res}(g(Y), f(X - Y), Y) \\
f \ominus g &= \mathrm{Res}(g(Y), f(X + Y), Y) \\
f \otimes g &= \mathrm{Res}(g(Y), Y^n f(X/Y), Y) \\
f \oslash g &= \mathrm{Res}(g(Y), f(XY), Y) \\
f^{(k)} &= \mathrm{Res}(g(Y), X - Y^k, Y)
\end{aligned}$$

Theorem 4.5 is related to determining the minimal polynomials of algebraic numbers. See [22] or the exercises of Chapter 6 in [16].

Corollary 4.6. *Let $\mathbb{F}_{p^k}$ be a finite field, and let f and g be polynomials over $\mathbb{F}_{p^k}$ of degree n and m.*

- *The operations $f \oplus g, f \ominus g, f \otimes g, f \oslash g$ can be computed in $O(k^2(n+m)^5)$ $\mathbb{F}_p$-operations.*
- *The operation $f^{(k)}$ can be computed in $O(k^2 n^5)$ $\mathbb{F}_p$-operations.*

Proof. For the first statement we need to compute the resultant of two polynomials of degree n and m. In this case the entries of the Sylvester matrix are polynomials of degree less than $n + m$. Hence $O(k^2(n+m)^5)$ $\mathbb{F}_p$-operations are needed. For computing $f^{(k)}$ efficiently, we can compute $\alpha = X^k$

(mod $\langle f, p\rangle$) in $O(n^3 \log(k))$ $\mathbb{F}_p$-operations. Then $M(\alpha) = f^{(k)}$ and this by Theorem 4.4 this can be computed in $O(k^2 n^5)$ $\mathbb{F}_p$-operations. □

4.2 Polynomials of Odd Degree

In this section we give two algorithms for factoring odd degree polynomials over $\mathbb{F}_p$ with a certain property deterministically and unconditionally and prove their correctness. These are Algorithm 8 and Algorithm 9. At the end of this section we apply these algorithms to the factorization of random polynomials over $\mathbb{F}_p$.

Let $s = \nu_2(p-1)$ and $m = (p-1)/2^s$. Algorithm 8 factors any polynomial f of odd degree so that $X^m \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$. Algorithm 8 makes use of the following algorithms:

- $M(\beta)$: let $R = \mathbb{F}_p[X]/(f)$ and $\beta \in R$ then $M(\beta)$ is the minimal polynomial of β
- BASE : given $f = f_1^{e_1} \cdots f_k^{e_k}$ returns $f_1 \cdots f_k$
- NON_FLAT : Algorithm 7

Algorithm 8: ODD_DEG

Input : p, f
  p a prime
  $f \in L_n$ so that f is squarefree
Output : FAIL if f cannot be factored, otherwise a nontrivial factor of f

$n = \deg(f)$
$s = \nu_2(p-1)$
$m = (p-1)/2^s$
$\alpha = X^m \pmod{\langle f, p\rangle}$
if $\alpha \in \mathbb{F}_p$ **then**
  Return FAIL
end
while $\deg(\alpha) \neq 0$ **do**
  $y = \alpha \pmod{\langle f, p\rangle}$
  $\alpha = \alpha^2 \pmod{\langle f, p\rangle}$
end
/* Compute the minimal polynomial of y. This polynomial has 2 distinct roots in $\mathbb{F}_p$ */
$v = M(y)$
$b = \text{BASE}(v, p)$
$v = \text{NON\_FLAT}(v, b, p)$
/* r is a root of v. We have that r is a square root of α in $\mathbb{F}_p$ */
$r = -v(0)$
Return $\gcd(y - r, f)$

For example, let $p = 113$, and $f = X^3 + 53X^2 + 83X + 35$ over $\mathbb{F}_p$. As $113 - 1 = 7 \cdot 2^4$ we have $s = 4$. Then $\alpha = 42X^2 + 79X + 96$, $\alpha^2 = 99X^2 + 43X + 66$, and $\alpha^4 = 112$. The components of $\alpha, \alpha^2, \alpha^4$ are 16th, 8th, and 4th roots of unity respectively. We note that none of the roots of f are primitive 2^s roots of unity as α^4 has all its components equal to $-1 \pmod{p}$. We let $y = 99X^2 + 43X + 66$. Then

$v = X^3 + 98X^2 + X + 98$ and $b = X^2 + 1$. Applying NOT_FLAT to v gives the factor $X - 98$ of v so $r = 98$ is a square root of -1 in $\mathbb{F}_p$. Finally, $\gcd(y - 98, f) = X + 48$ is a factor of f.

Theorem 4.7. *Let $f \in L_n$ for n odd, and let m, s be such that $p - 1 = m2^s$. If $X^m \pmod f \notin \mathbb{F}_p$ then Algorithm 8 factors f deterministically in $O(\log(p)n^2 + n^5)$ operations.*

Proof. The first while loop terminates with y such that $\deg(y) > 0$ and $\zeta = y^2 \pmod{\langle f, p\rangle}$ is in $\mathbb{F}_p$. So y has at least one distinct component. On the other hand $(y - \sqrt{\zeta})(y + \sqrt{\zeta}) = 0$ so the distinct components of y are $\sqrt{\zeta}$ and $-\sqrt{\zeta}$ and so $y = (\pm\sqrt{\zeta}, \ldots, \pm\sqrt{\zeta})$. As the degree of f is odd we cannot have $\sqrt{\zeta}$ and $-\sqrt{\zeta}$ appear the same number of times as components of y. Let k_1 and k_2 be the number of times that $\sqrt{\zeta}$ and $-\sqrt{\zeta}$ are components of y respectively. Then $k_1 > 0, k_2 > 0$ and the minimal polynomial m of y is $(X - \sqrt{\zeta})^{k_1}(X + \sqrt{\zeta})^{k_2}$. As m is not flat we can find a factor of m. Hence we can find a root $r = \sqrt{\zeta}$ of v. The element $y - r$ is zero on at least one component but not on every component.

Computation of $X^m \pmod{\langle f, p\rangle}$ takes $O(\log(m)n^2) = (\log(p)n^2)$ operations. Within the first while loop, computation of a square of α takes $O(n^2)$ operations. Hence computation of all the squares takes $O(\log(m)n^2)$. Computing BASE and NON_FLAT both take $O(n^3)$ operations. Computation of the minimal polynomial is done by computing a single resultant which takes $O(n^5)$ operations. □

We note that Algorithm 8 can also be applied to polynomials f of even degree. However, even if $X^m \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$ there is no guarantee that the algorithm will produce a factor f.

We now turn to the second algorithm for factoring polynomials. This algorithm makes use of the composed operations $\otimes$ and $\oslash$. These operations can be used as a tool for FPFF in the following way. Given a polynomial f we may compute $t = f \oslash f$. We then attempt a factorization of t with some polynomial factorization algorithm $\mathcal{A}$. If $\mathcal{A}$ produces a factor g of t then we compute $r = g \otimes f$. The polynomial r has at least one root in common with f. Whenever r is not flat with respect to f then a factor of f can be computed in deterministic polynomial time.

For example, if $f \in L_3$ over some finite field with roots r_1, r_2, r_3 then $h = (X - r_1/r_2)(X - r_1/r_3)(X - r_2/r_1)(X - r_2/r_3)(X - r_3/r_1)(X - r_3/r_2)$. Suppose the algorithm $\mathcal{A}$ returns $s = (X - r_1/r_2)(X - r_1/r_3)$ when given h as input. Then

$$\begin{aligned} r = s \otimes f &= \big(X - r_1^2/r_2\big)\big(X - r_1^2/r_3\big)\big(X - r_1\big)\big(X - r_1r_2/r_3\big)\big(X - r_1r_3/r_2\big)\big(X - r_1\big) \\ &= \big(X - r_1\big)^2\big(X - r_1^2/r_2\big)\big(X - r_1^2/r_3\big)\big(X - r_1r_2/r_3\big)\big(X - r_1r_3/r_2\big). \end{aligned}$$

Intuitively speaking we do not expect r to be flat because this would require equality between one of $\{r_1^2/r_2, r_1^2/r_3, r_1r_2/r_3, r_1r_3/r_2\}$ and r_2.

Definition 4.8. Let $f \in L_n$ and let R be the set of roots of f, γ be an arbitrary element of R, and $t_1 = (X-1)^{-n}(f \oslash f)$ then a factor h of t_1 is $\otimes$-*balanced* with respect to f if for each root $a \in R$

$$\#\{b \in R : h(a/b) = 0\} = \#\{b \in R : h(\gamma/b) = 0\}.$$

Suppose $t_1 = h$. Clearly t_1 is balanced as each root appears the same number of times in the numerator and denominator of the roots of h. The number of numerator positions is $n(n-1)$ and the number of denominator positions is $n(n-1)$. The number of roots is n so we have

$$\#\{b \in R : h(a/b) = 0\} = n - 1 \tag{1}$$

The degree of h is sometimes enough to show h is unbalanced. If $m = \deg(h)$ and if $n \nmid m$ then h is not $\otimes$-balanced. As an example of a balanced polynomial, let $f = (X - r_1)(X - r_2)(X - r_3)$ then $t_1 = (X - r_1/r_2)(X - r_1/r_3)(X - r_2/r_1)(X - r_2/r_3)(X - r_3/r_1)(X - r_3/r_2)$. Let $h = t_1$ then for $\gamma = r_1, r_2, r_3$, respectively,

$$\begin{aligned}
\#\{b \in R : h(r_1/b) = 0\} &= \#\{r_2, r_3\} = 2 \\
\#\{b \in R : h(r_2/b) = 0\} &= \#\{r_1, r_3\} = 2 \\
\#\{b \in R : h(r_3/b) = 0\} &= \#\{r_1, r_2\} = 2
\end{aligned}$$

so h is balanced.

As a second example, let $f = (X-r_1)(X-r_2)(X-r_3)(X-r_4)$ then $h = (X-r_1/r_2)(X-r_2/r_1)(X-r_3/r_4)(X-r_4/r_3)$ is balanced.

We may view polynomials that are factors of $(X-1)^{-n}(f \oslash f)$ as the arc polynomial of a graph. From this point of view, balanced polynomials correspond to regular subgraphs.

For example, let $f = (X - r_1)(X - r_2)(X - r_3)(X - r_4)$ and consider the arc polynomial

$$h = \left(X - \frac{r_1}{r_2}\right)\left(X - \frac{r_2}{r_1}\right)\left(X - \frac{r_2}{r_3}\right)\left(X - \frac{r_3}{r_2}\right)\left(X - \frac{r_3}{r_4}\right)\left(X - \frac{r_4}{r_3}\right)\left(X - \frac{r_4}{r_1}\right)\left(X - \frac{r_1}{r_4}\right).$$

Then Figure 2 gives the corresponding directed graph.

Lemma 4.9. *Given polynomials $f \in L_n$ let $t = (X-1)^{-n}(f \oslash f)$. If g is a balanced factor of t with respect to f then t/f is also balanced.*

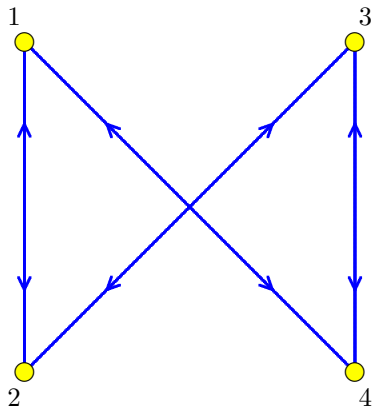

Figure 2: The graph of a balanced arc polynomial.

Proof. Let $r_1, \ldots, r_n$ be the roots of f. Then

$$\begin{aligned} t =&(X - r_1/r_2)(X - r_1/r_3)\ldots(X - r_1/r_n) \\ &(X - r_2/r_1)(X - r_2/r_3)\ldots(X - r_2/r_n) \\ &\vdots \\ &(X - r_n/r_1)(X - r_n/r_3)\ldots(X - r_n/r_{n-1}) \end{aligned}$$

Let $h = t/g$ then for any root a of f we have

$$n - 1 = \#\{b \in R : g(a/b) = 0\} + \#\{b \in R : h(a/b) = 0\} = \#\{b \in R : g(a/b) = 0\} + k$$

for some positive integer k. Hence

$$\#\{b \in R : g\Big(\frac{a}{b}\Big) = 0\} = n - 1 - k$$

which shows that g is balanced. □

Our strategy for polynomial factorization is as follows. Let $f \in L_n$ and $t = (X-1)^{-n}(f \oslash f)$. If we can give a sufficient condition under which 1) any factor g of t that is unbalanced with respect to f is such that $g \otimes f$ is not flat, and 2) we are given a polynomial factorization algorithm $\mathcal{A}$ then we can construct a polynomial factorization algorithm $\mathcal{B}$ that factors any polynomial f for which $\mathcal{A}$ factors t.

Lemma 4.10. *Let $f \in L_n$ and $t = (X-1)^{-n}(f \oslash f)$ is squarefree. If r_1 and r_2 are any roots of f then the only solution to $(r_1/r_2)X = Y$ for X and Y roots of f is $X = r_2$ and $Y = r_1$.*

Proof. Suppose there exist s_1, s_2 such that $s_1 \neq r_2$ or $s_2 \neq r_1$ such that $(r_1/r_2)s_1 = s_2$ then $(r_1/s_2) =$

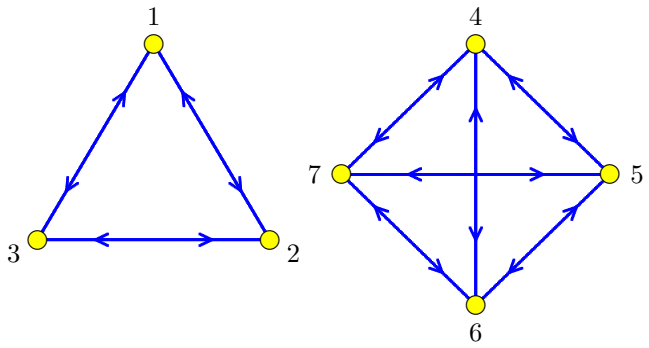

Figure 3: Difference of indexes equal to zero.

(s_2/r_1). But (r_1/s_2) is a root of t which contradicts the fact that t is squarefree. □

Corollary 4.11. *Let $f \in L_n$ such that $t = (X-1)^{-n}(f \oslash f)$ is squarefree, R be the set of roots of f, and h be a factor of $(X-1)^{-n}(f \oslash f)$. Then $h \otimes f$ is flat if and only if h is balanced.*

Proof. Let R be the set of roots of f. Then the roots of $h \otimes f$ have the form ac/b where $a, b, c \in R$.

Let $T(r) = \{b \in R : h(r_1/b) = 0\}$ and let $N(r) = \{s_1, s_2 \in R, r/b \in S : (r/b)s_1 = s_2\}$. By Lemma 4.10 the only solution to $(r_1/r_2)X = Y$ for X and Y roots of f is $X = r_2$ and $Y = r_1$ so $\#T(r) = \#N(r)$.

If f is balanced then for any pair $r_1, r_2 \in R$ we have $\#N(r_1) = \#T(r_1) = \#T(r_2) = \#N(r_2)$ so $h \otimes f$ is flat. If f is not balanced then their exist $r_1, r_2 \in R$ such that $\#N(r_1) = \#T(r_1) \neq \#T(r_2) = \#N(r_1)$ so $h \otimes f$ is not flat. □

Before giving Theorem 4.12 we give an example to motivate its proof. Suppose we are given the polynomial $f = (X-r_1)(X-r_2)(X-r_3)(X-r_4)(X-r_5)(X-r_6)(X-r_7)$ where $r_1, \ldots, r_7 \in \mathbb{F}_p$ such that $p \equiv 3 \pmod 4$ and

$$\text{ind}(r_1) \equiv \text{ind}(r_2) \equiv \text{ind}(r_3) \equiv 0 \pmod 2 \qquad \text{ind}(r_4) \equiv \text{ind}(r_5) \equiv \text{ind}(r_6) \equiv \text{ind}(r_7) \equiv 1 \pmod 2.$$

There are $7 \cdot 6 = 42$ pairs of distinct roots. Let $v_{i,j} = \text{ind}(r_i) - \text{ind}(r_j) \pmod 2$. We can form two subgraphs G_0, G_1 of the complete graph on vertices $1, \ldots, 7$ where r_i corresponds to vertex i and G_0 has the arc $[i, j]$ when $v_{i,j} = 0$ and G_1 has the arc $[i, j]$ when $v_{i,j} = 1$. See Figure 3 and Figure 4.

An interesting property of these graphs is that they are not regular. In the graph in Figure 4 the vertices $1, 2, 3$ have 4 arcs in and out, and the other vertices have 3 arcs in and out. The arc polynomials of G_0

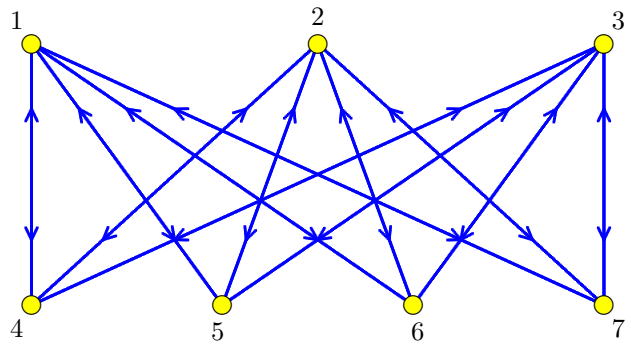

Figure 4: Difference of indexes equal to one.

and G_1 with respect to multiplication are $g_0 = t_0 t_0^*$ and $g_1 = t_1 t_1^*$ where

$$\begin{aligned} t_0 =& \left(X - \frac{r_1}{r_2}\right)\left(X - \frac{r_1}{r_3}\right)\left(X - \frac{r_2}{r_3}\right)\left(X - \frac{r_4}{r_5}\right)\left(X - \frac{r_4}{r_6}\right)\left(X - \frac{r_4}{r_7}\right)\left(X - \frac{r_5}{r_6}\right) \\ & \cdot \left(X - \frac{r_5}{r_7}\right)\left(X - \frac{r_6}{r_7}\right), \\ t_1 =& \left(X - \frac{r_1}{r_4}\right)\left(X - \frac{r_1}{r_5}\right)\left(X - \frac{r_1}{r_6}\right)\left(X - \frac{r_1}{r_7}\right)\left(X - \frac{r_2}{r_4}\right)\left(X - \frac{r_2}{r_5}\right)\left(X - \frac{r_2}{r_6}\right) \\ & \cdot \left(X - \frac{r_2}{r_7}\right)\left(X - \frac{r_3}{r_4}\right)\left(X - \frac{r_3}{r_5}\right)\left(X - \frac{r_3}{r_6}\right)\left(X - \frac{r_3}{r_7}\right). \end{aligned}$$

Let $t = (X-1)^{-n}(f \oslash f)$, $m = (p-1)/2$, and $\alpha = X^m \pmod f$ then by Lemma 2.32 $g_0 = \gcd(\alpha - 1, t)$ and $g_1 = \gcd(\alpha + 1, t)$. The polynomials t_0 and t_1 are not balanced so we can construct a factor of f. This idea is naturally extended to factoring a large class of odd degree polynomials by Theorem 4.12.

We recall some notation from Chapter 2 that is used in the proof of Theorem 4.12. We defined

$$\text{ind}(f) = \{\text{ind}(r_1), \ldots, \text{ind}(r_n)\}$$

where $\text{ind}(r_1), \ldots, \text{ind}(r_n)$ are computed with respect to g.

For $x \in \mathbb{F}_q$ and for l a prime dividing $q-1$ define $\text{ind}(x)[l^k]$ be the kth digit of $\text{ind}(x)$ expressed in base l with the lowest digit defined as the 0th digit. For $f \in L_n$ with roots $r_1, \ldots, r_n$ define

$$\text{ind}(f)[l^k] = \{\text{ind}(r_1)[l^k], \ldots, \text{ind}(r_n)[l^k]\}.$$

Like Algorithm 8, Algorithm 9 factors some polynomials of odd degree over prime fields. There are more conditions required for Algorithm 9 to guarantee that a factor is found than for Algorithm 8. However, Algorithm 9 is related to a method used in the next chapter. Algorithm 9 makes use of the following algorithms:

- NON_FLAT : Algorithm 7

- IS_SQF : Takes a polynomial f as input and returns true if f is squarefree and false otherwise

Algorithm 9: ODD_DEG2

Input : p, f
p a prime
f a squarefree polynomial in L_n over $\mathbb{F}_p$
Output : if $X^m \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$ and $(X-1)^{-n}(f \oslash f)$ is squarefree returns a nontrivial factor of f

$s = \nu_2(p-1)$
$m = (p-1)/2^s$
$\alpha = X^m \pmod{\langle f, p\rangle}$
if $\alpha \in \mathbb{F}_p$ **then**
`/* in this case all the components of` α `are` m`th roots of unity */`
Return FAIL
end
else
$t = (X-1)^{-n}(f \oslash f)$
if *IS_SQF*(t, p)=*false* **then** Return FAIL

$z = X^m \pmod{\langle t, p\rangle}$
while $z \notin \mathbb{F}_p$ **do**
$y = z$
$z = z^2 \pmod{\langle t, p\rangle}$
end
$g_1 = \gcd(y-1, t)$
$b = g_1 \otimes f$
$f_1 = \text{NON\_FLAT}(b, f, p)$
Return f_1
end

We illustrate the Algorithm 9. Let $\mathbb{F}_p$ be the prime field with $p = 1033$ then $p-1 = m \cdot 2^3$ for $m = 129$ and $r_1 = 33, r_2 = 67, r_3 = 308, r_4 = 327, r_5 = 409$ and $f = (X-r_1)(X-r_2)(X-r_3)(X-r_4)(X-r_5)$. Clearly f is squarefree. We let $\alpha = X^m \pmod f$. The element $z \notin \mathbb{F}_p$. However, $z^2 = 355$ and so $z = 355$ is a 4th primitive root of unity. This tells us that

$$\text{ind}(f) = [\text{ind}(r_1), \text{ind}(r_2), \text{ind}(r_3), \text{ind}(r_4), \text{ind}(r_5)] \equiv k[1,1,1,1,1] \pmod{2^2}$$

for some integer k and there exist $1 \le i < j \le 5$ so that $\text{ind}(r_i) \not\equiv \text{ind}(r_j) \pmod{2^3}$. Let $t = (X-1)^{-5}(f \oslash f)$. We have

$$t = X^{20} + 180X^{19} + 21X^{18} + 54X^{17} + 598X^{16} + 555X^{15} + 228X^{14} + 915X^{13} + 393X^{12} + 548X^{11}$$
$$+ 119X^{10} + 548X^9 + 393X^8 + 915X^7 + 228X^6 + 555X^5 + 598X^4 + 54X^3 + 21X^2 + 180X + 1.$$

We verify that t is squarefree. Then $z = X^m \pmod t \notin \mathbb{F}_p$ and $z^2 = 1$ so by Theorem 2.32 $z-1$ splits

t. Let $h_0 = \gcd(z-1, t)$ and $h_1 = \gcd(z+1, t)$. We have

$$\begin{aligned} h_0 &= X^8 + 652X^7 + 888X^6 + 444X^5 + 143X^4 + 444X^3 + 888X^2 + 652X + 1, \\ h_1 &= X^{12} + 561X^{11} + 76X^{10} + 413X^9 + 318X^8 + 42X^7 + 338X^6 + 42X^5 + 318X^4 + 413X^3 \\ &\quad + 76X^2 + 561X + 1. \end{aligned}$$

We know that the first of these factors corresponds to all the pairs of roots r_i, r_j for which the difference $\text{ind}(r_i)[2^2] - \text{ind}(r_j)[2^2] \equiv 0 \pmod 2$ and the second corresponds to all the pairs of roots r_i, r_j for which the difference $\text{ind}(r_i)[2^2] - \text{ind}(r_j)[2^2] \equiv 1 \pmod 2$. As the degree of h_1 is not divisible by n we have that h is not balanced. Then $r = h_1 \otimes f$ is a degree 60 polynomial. Applying Algorithm 7 to this polynomial we recover the factor $X^2 + 591X + 68 = (X - 33)(X - 409) = (X - r_1)(X - r_5)$.

We can give a bit more detail about the roots of the factor we have recovered. We have $g = 5$ is a primitive root in $\mathbb{F}_p$. Then $\text{ind}(f)$ with respect to g is $[\text{ind}(r_1), \text{ind}(r_2), \text{ind}(r_3), \text{ind}(r_4), \text{ind}(r_5)] = [579, 263, 327, 871, 595]$ and $\text{ind}(f) \equiv [3, 7, 7, 7, 3] \pmod{2^3}$. Hence we see that the roots of f that are in the factor $X^2 + 591X + 68$ have index that is congruent to 3 $\pmod{2^3}$. In the arc polynomial h_1 the vertices corresponding to r_1 and r_5 have three arcs in and three arcs out. These arcs join them the vertices r_2, r_3, r_4. Each of the vertices r_2, r_3, r_4 have two arcs in and two arcs out. These arcs join them to r_1 and r_5. As r_1 and r_5 have more arcs in arcs out than the vertices r_2, r_3, r_4 Algorithm 7 returns $(X - r_1)(X - r_5)$. This concludes the example.

Theorem 4.12. *Let $f \in L_n$ for n odd, and $t = (X-1)^{-n}(f \oslash f)$, and let m be such that $p - 1 = m2^s$. If t is squarefree, and $X^m \pmod f \notin \mathbb{F}_p$ then Algorithm 9 factors f deterministically in $\text{poly}(n, \log(p))$ operations. More precisely the algorithm uses $O(n^{10}\log(p)^2 + n^4\log(p)^3)$ operations.*

Proof. Let $t = (X-1)^{-n}(f \oslash f)$. The degree of t is $n(n-1)$. Consider $\text{ind}(f) \equiv (\text{ind}(r_1), \ldots, \text{ind}(r_n)) \pmod{2^s}$. Let i be the largest positive integer so that for every pair of roots r_a, r_b we have $\text{ind}(r_a) \equiv \text{ind}(r_b) \pmod{2^i}$. We have $i < s$ since $X^m \pmod f \notin \mathbb{F}_p$. Then there is a pair of roots r_a, r_b so that $\text{ind}(r_a) - \text{ind}(r_b) \equiv 0 \pmod{2^{i+1}}$ and a pair of roots r_c, r_d so that $\text{ind}(r_c) - \text{ind}(r_d) \not\equiv 0 \pmod 2^{i+1}$. It follows that the order of $\text{ind}(r_c) - \text{ind}(r_d)$ is divisible by a larger power of 2 than the order of $\text{ind}(r_a) - \text{ind}(r_b)$ so by Theorem 2.32 we can get a factorization $t = g_0 g_1$. Let

$$\begin{aligned} E_0 &= \{[r_i, r_j] \in R : \text{ind}(r_i)[2^{i+1}] - \text{ind}(r_j)[2^{i+1}]\} \\ E_1 &= \{[r_i, r_j] \in R : \text{ind}(r_i)[2^{i+1}] - \text{ind}(r_j)[2^{i+1}]\}. \end{aligned}$$

Then $g_0 = \prod_{[r_i,r_j]\in E_0}(X - r_i/r_j)$ and $g_1 = \prod_{[r_i,r_j]\in E_1}(X - r_i/r_j)$. We show neither of these is balanced

by showing g_1 is not balanced. Let $R_0 = \{r \in R : (\text{ind}(r))[2^{i+1}] \equiv 0 \pmod 2\}$ and $R_1 = \{r \in R : (\text{ind}(r))[2^{i+1}] \equiv 1 \pmod 2\}$. These sets are a disjoint partition of R and since n is odd they do not have equal size. Every element of R_0 appears $(\#R_1)$ times in the first position of the elements of E_1 and $(\#R_2)$ times in the second position of the elements of E_2. Every element of R_1 appears $(\#R_0)$ times in the first position of the elements of E_0 and $(\#R_1)$ times in the second position of the elements of E_0. Hence g_1 is not balanced and so by Corollary 4.11 and Algorithm 7 a nontrivial factor of f can be constructed in deterministic polynomial time.

We now turn to the number of operations. We have α can be computed in $O(n^2 \log(p)^3)$ by Theorem 2.29. Then t can be computed in $O(n^3 T(n)) = O(n^5)$ operations by Theorem 4.6 where $T(n)$ is the number of operations to compute the product of two polynomials of degree n. As t is a degree n^2 polynomial it can be tested if it is squarefree in $O(n^4 \log(p)^2)$ by Theorem 2.29. Then z can be computed in $O(n^4 \log(p)^3)$ operations by Theorem 2.29 and g_1 can be computed in $O(n^4 \log(p)^2)$. Using Theorem 4.6 we can compute b in $O(n^{10} \log(p)^2)$ operations by Theorem 4.6 and f_1 can be computed in $O(n^8 \log(p)^2)$ by Theorem 2.37.

□

We remark that if we are allowed to use ERH then Theorem 4.12 is more easily proved. By Theorem 3.5 a quadratic nonresidue can be constructed and by Theorem 2.35 f can be factored in deterministic polynomial time. Note that an algorithm to factor *arbitrary* odd degree polynomials in L_n under ERH would imply that any polynomial over $\mathbb{F}_p$ can be factored in deterministic polynomial time by Theorem 3.7 (for generic polynomials this theorem requires GRH to produce a factor, however, for the case that n is even we only need to construct a quadratic nonresidue so ERH is sufficient).

Although so far we have restricted our attention to the case that n is odd Algorithm 8 can also be run on even degree polynomials. In this case the conditions of the odd degree case do not guarantee that a factor is produced. However, intuitively speaking, the algorithm in the proof of Algorithm 8 factors many polynomials of both odd and even degree. The following theorem shows the probability the algorithm produces a factor of random element of L_n is better than the probability that the Algorithm 3 produces a factor of a random element of L_n modulo some primes. In particular, it is better when $p \equiv 1 \pmod 4$.

For any $f \in L_n$ such that $f^{(p-1)/2} \notin \mathbb{F}_p$ then f can be factored unconditionally by Theorem 2.32. This kind of factorization is a special case of Algorithm 3. If we let M be the subset of L_n for which Algorithm 3 algorithm fails to output a factor then the probability that a random element of L_n is in M is $1/2^{n-1}$ since factorization occurs when every root is a quadratic residue or every root is a quadratic nonresidue. A random element of $\mathbb{F}_p^*$ is a quadratic nonresidue with probability $1/2$ by Corollary 2.22.

Lemma 4.13. *Let n be a positive integer then $\binom{2n}{n} \leq 4^n$.*

Proof. From Stirling's approximation we have $n! \leq en^{n+1/2}e^{-n}$. Then

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \leq \frac{e(2n)^{2n+1/2}e^{2n}}{e^2(2n)^{2n+1}e^{2n}} = \frac{4^n}{\sqrt{2}en^{1/2}} \leq 4^n$$

□

Theorem 4.14. *Let $\mathcal{A}$ denote Algorithm 8. Let S be the set of polynomials in L_n so $\mathcal{A}$ fails to produce a factor. Suppose $p-1 = m2^s$ for odd m. The probability that a random element of L_n is in S is less than $\frac{1}{2^{n(s-1)}}$.*

Proof. Let $P(f \in S)$ be the probability that f is in S. We have

$$P(f \in S) = P(f^m \notin \mathbb{F}_p)P(f \in S \mid f^m \notin \mathbb{F}_p).$$

We form the $n \times s$ array B with entries in $\{0,1\}$ where $b_{i,j} = \text{ind}(r_i)[2^j]$. Then $f^m \notin \mathbb{F}_p$ if and only if there is a column of B not equal to $k[1,\ldots,1]$ for $k \in \{0,1\}$. For any $0 \leq z \leq 2^s - 1$ there are an equal number of elements in the set $\{x \in \mathbb{F}_p : \text{ind}(x) \equiv z \pmod{2^s}\}$ so the probability that a given column of B has the form $k[1,\ldots,1]$ for $k \in \{0,1\}$ is $1/2^n$ and the probability that every column of B has this form is $1/2^{ns}$.

Given that $f^m \notin \mathbb{F}_p$ let i be the least positive integer so that two entries in the ith column are distinct. Then $\mathcal{A}$ only fails to produce a factor of f if the ith column has the same number of zeroes as it does ones. Hence by 4.13

$$P(f \in S \mid f^m \notin \mathbb{F}_p) = \binom{2n}{n}2^{-n} \leq 4^n 2^{-n} = 2^n$$

Then $P(f \in S) = P(f^m \notin \mathbb{F}_p)P(f \in S \mid f^m \notin \mathbb{F}_p) \leq \frac{2^n}{2^{ns}} = \frac{1}{2^{n(s-1)}}$. □

4.3 Factoring Polynomials with $\gcd(n, \phi(p-1)) = 1$

Suppose we are given the polynomial $f = (X-\omega)(X-\omega^2)(X-\omega^4)$ to factor modulo a prime p in coefficient form with ω a primitive 5th root of unity. We can try to factor this polynomial using the composed power operation. We compute $f^{(2)} = (X-\omega^2)(X-\omega^4)(X-\omega^3)$ and as two of the roots of this polynomial are roots of f we can factor f by computing its greatest common divisor with $f^{(2)}$.

A more general version of this idea would be to compute $h_k = \gcd(f, f^{(k)})$ for $k = 1, 2, \ldots$ until h_k is a nontrivial factor of f. It is clear that $h_1, h_2, \ldots$ is a periodic sequence and the period divides $p-1$.

We want to construct the set of all polynomials of this type that cannot be factored by the method above. Consider the case where f is a polynomial in L_n with roots that are distinct lth primitive roots of unity for prime l. We note that $\text{ind}(f)$ is a subset of $\mathbb{Z}^*_{l-1}$.

Let C_d denote the multiplicative cosets of $\mathbb{Z}_{l-1}^*$. Then the elements of C_d form a partition of $\mathbb{Z}_{l-1}^*$. If $k \in \mathbb{Z}_{l-1}^*$ and c_1, c_2 are elements of the same coset then kc_1 and kc_2 are in the same coset. Let $\text{ind}(f)[l] = [z_0, \ldots, z_n]$ be a multiplicative coset of a subgroup of $\mathbb{Z}_l^*$. For any positive integer c that is not divisible by l we have $\text{ind}(f^{(c)})[l] = [cz_0, \ldots, cz_n]$ is also a coset of H. Hence the polynomials f that we cannot factor by computing $h_k = \gcd(f, f^{(k)})$ for $k = 1, 2, \ldots$ are the polynomials where $\text{ind}(f)$ is a coset of $\mathbb{Z}_{l-1}^*$.

Heuristically, the least $k > 2$ such that $\gcd(f, f^{(k)}) > 1$ could be large. Algorithm 11 improves on the efficiency of this idea. In Theorem 4.15 it is shown that any squarefree polynomial in L_n with $\gcd(n, \phi(p-1)) = 1$ is factored by Algorithm 11. As in Theorem 3.12 the number of operations of this algorithm depends on $S(p-1)$ but Algorithm 11 is unconditionally deterministic, unlike Theorem 3.12 which is conditional on ERH.

Recall that if f is an element of L_n over $\mathbb{F}_p$ then f is a univariate polynomial in $\mathbb{F}_p[X]$. Algorithm 10 is an auxiliary algorithm for Algorithm 11. Algorithm 10 makes use of the following algorithms:

- $M(\alpha)$: the minimal polynomial of α
- IS_SQF : returns true if a polynomial is squarefree and false otherwise

Algorithm 10: PRIME_POW_FAC

Input: p, l, f
p a prime
l a prime dividing $p-1$ so that $X^{(p-1)/l^s} \notin \mathbb{F}_p$
$f \in L_n$, so that f is squarefree and $\gcd(n, \phi(p-1)) = 1$
Output: a factor of $f^{(de)}$ with d, e as defined below

$s = \nu_l(p-1)$
$d = (p-1)/l^s$
$e = d^{-1} \pmod{l^s}$
$\alpha = X^{(de)} \pmod{\langle f, p \rangle}$
/* $M(\alpha)$ is equal to $f^{(de)}$ */
$g = M(\alpha)$
/* initialize A as a tuple with g as its only element */
$A = [g]$
/* compute the minimal polynomial of α and test if it is squarefree */
while $IS\_SQF(g, \mathbb{F}_p)$ **do**
    $\beta = \alpha$
    $\alpha = \alpha^l \pmod{\langle f, p \rangle}$
    $g = M(\alpha)$
    /* add the element m to the tuple A */
    $A = A \cup g$
end
/* when we exit the while loop A is a tuple of z elements $[A_1, \ldots, A_z]$ and $\deg(M(\beta)) = n$ */
$z = \#A$
$v = M(\beta)$
/* c is a product of the roots of v */
$c = (-1)^n v(0)$
$b = n^{-1} \pmod{l^s}$
/* the constant y is used to manipulate the roots of the polynomial v */
$y = c^b$
/* by evaluating v at yX make a new polynomial so that */
/* for every root r of v we have $y^{-1}r$ is a root of the new polynomial */
$v = v(yX)$
/* test if $X = 1$ is a root modulo p. In this case $X - y$ is a root of m */
$r = v(1)$
if $r = 0$ **then**
    $h = X - y$;
for i **from** 2 **to** $l - 1$ **do**
    $w = v^{(i)}$
    $h = \gcd(v, w)$
    if $0 < \deg(h)$ *and* $\deg(h) < n$ **then**
        Break
end
if $\deg(h) = 0$ *or* $\deg(h) = n$ **then**
    /* in this case $\mathrm{ind}(h)$ is a coset of $\mathbb{F}_l^*$ */
    Return FAIL
/* h is a nontrivial factor of A_{z-1}. We divide the constant y out from the roots of h */
$h = h(y^{-1}X)$
for i **from** 2 **to** $z - 1$ **do**
    $h = h(X^l)$
    $h = \gcd(a_{z-i}, h)$
end
Return h

We note that Algorithm 10 is similar to Algorithm 4 and Algorithm 6. Each of these algorithms requires repeatedly raising an element α to the power l until α is in $\mathbb{F}_p$. For each such α we store an lth root of α that is not in $\mathbb{F}_p$. The main difference between these algorithms is in how a factor of the polynomial f is found using the lth root of α.

Algorithm 11 factors polynomials with degree coprime to $\phi(p-1)$. Algorithm 11 makes use of the following algorithms:

- FIND_PRIME : finds the smallest prime l so that $\alpha^{(p-1)/l^s} \notin \mathbb{F}_p$ for $s = \nu_l(p-1)$
- PRIME_POW_FAC : Algorithm 10

Algorithm 11: CO_DEG

Input: p, f
p prime
$f \in L_n$, so that f is squarefree and $\gcd(n, \phi(p-1)) = 1$
Output: a factor of f

$l = \text{FIND\_PRIME}(f, p)$
$s = \nu_l(p-1)$
$d = l^s$
$e = d^{-1} \pmod{(p-1)/l^s}$
$t = f^{(de)}$
$h = \text{PRIME\_POW\_FAC}(f, l, p)$
$y = h \otimes t$
$a = \gcd(y, f)$
Return a

For example, let $p = 13751$, and $\omega = 15$ a 5^4th primitive root of unity. In order to make the example clearer we show polynomials in their factored form. Let $f = (X-\omega)(X-\omega^{3+5})(X-\omega^{4+5+5^2})$. The prime found by FIND_PRIME is 5. As all the roots are 5^4th primitive roots of unity the polynomial t in Algorithm 11 is $(X-1)^3$. The list of minimal polynomials A in Algorithm 10 is

$$\begin{aligned}[a_1, a_2, a_3, a_4] = &\big[\big(X-\omega\big)\big(X-\omega^{3+5}\big)\big(X-\omega^{4+5+5^2}\big),\\ &\big(X-\omega^{5}\big)\big(X-\omega^{3\cdot 5+5^2}\big)\big(X-\omega^{4\cdot 5+5^2+5^3}\big),\\ &\big(X-\omega^{5^2}\big)\big(X-\omega^{3\cdot 5^2+5^3}\big)\big(X-\omega^{4\cdot 5^2+5^3}\big),\\ &\big(X-\omega^{5^3}\big)\big(X-\omega^{3\cdot 5^3}\big)\big(X-\omega^{4\cdot 5^3}\big)\big].\end{aligned}$$

Let $Z = \text{ind}(a_4)[5^3] = \{1, 3, 4\}$. The least integer i so that Z and iZ have an intersection that is non-empty but does not equal Z is $i = 2$. In this case $Z \cap iZ = \{1, 3\}$. Hence $\gcd(a_4, a_4^{(2)}) = (X-\omega^{5^3})(X-\omega^{3\cdot 5^3})$. In the next iteration of the loop the polynomial g becomes $(X-\omega^{5^2})(X-\omega^{3\cdot 5^2+5^3})$ then $(X-\omega^5)(X-\omega^{3\cdot 5+5^2})$, and then $(X-\omega)(X-\omega^{3+5})$. Algorithm 10 returns this g. Then in Algorithm 11 we have $y = h \otimes t = ((X-\omega)(X-\omega^{3+5}))^3$. Finally, $a = \gcd(y, f) = (X-\omega)(X-\omega^{3+5})$ is a nontrivial factor of f.

Unfortunately there can be many polynomials of a given degree that this algorithm does not factor. For example let p be a prime so that $p-1$ is divisible by 7. Let ω be a generator of the subgroup of elements whose order is not divisible by 7 and ζ be a primitive seventh root of unity. Then the algorithm will fail to factor polynomials of the form

$$f = \big(X - \omega^i\zeta\big)\big(X - \omega^i\zeta^2\big)\big(X - \omega^i\zeta^4\big).$$

for any nonnegative integer i. Given this f the polynomial g in Algorithm 10 is

$$g = \big(X - \zeta\big)\big(X - \zeta^2\big)\big(X - \zeta^4\big).$$

The set $\{1, 2, 4\}$ is a multiplicative subgroup of $\mathbb{Z}_6^*$ so Algorithm 10 does not find a factor of it.

For x a positive integer let $\mathrm{rad}(x)$ denote the prime factors of x. Regarding the statement of Theorem 4.15 we note that $\gcd(n, \phi(p-1)) = 1$ if and only if $\gcd(n, \mathrm{rad}(\phi(p-1))) = 1$.

Theorem 4.15. *Let p be a prime, and f be a squarefree polynomial in L_n. If $\gcd(n, \phi(p-1)) = 1$ then Algorithm 11 factors f deterministically in $O(n^5 \log(p) S(p-1))$ operations.*

Theorem 4.15 is a direct result of Theorem 4.16 and Lemma 4.17.

Theorem 4.16. *Let p be a prime, and f be a squarefree polynomial in L_n. Let l be a prime dividing $p-1$, $s = \nu(q-1)$, $d = (p-1)/l^s$. If $\gcd(n, \phi(p-1)) = 1$ and $X^d \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$ then Algorithm 10 factors f deterministically in $O(n^5 \log(p) S(p-1))$ operations.*

Proof. Let $p = l_1^{s_1} \cdots l_k^{s_k}$. The condition $\gcd(n, \phi(p-1)) = 1$ implies

$$\gcd(n, \mathrm{rad}(\phi(p-1))) = \gcd\left(n, \Big(\prod_{i=1}^{n} l_i\Big)\Big(\prod_{i=1}^{n} (l_i - 1)\Big)\right) = 1.$$

Hence n is coprime to l and $l-1$ for each prime l dividing $p-1$. In Algorithm 10 we start with a general polynomial f of degree n.

$$\begin{aligned}
\mathrm{ind}(f) = \big[a_{0,0} &+ a_{0,1}l &&+ \cdots + a_{0,s-1}l^{s-1} \\
a_{1,0} &+ a_{1,1}l &&+ \cdots + a_{1,s-1}l^{s-1}, \\
&\vdots &&\vdots \quad \vdots \\
a_{n-1,0} &+ a_{n-1,1}l &&+ \cdots + a_{n-1,s-1}l^{s-1}\big].
\end{aligned}$$

Let i be the least positive integer so that $\mathrm{ind}(f)[l^i]$ is a nonzero tuple. Using a composed power we

construct the polynomial v of degree n from f. The roots of v are l^ith primitive roots of unity for some i and the index of v has the form

$$\begin{aligned}\operatorname{ind}(v) = \big[&w_i l^i + w_{i+1} l^{i+1} + \cdots + w_{s-2} l^{s-2} + z_0 l^{s-1},\\ &w_i l^i \ + w_{i+1} l^{i+1} + \cdots + w_{s-2} l^{s-2} + z_1 l^{s-1},\\ &\vdots \qquad\quad \vdots \quad \vdots \qquad\quad \vdots\\ &w_i l^i \ + w_{i+1} l^{i+1} + \cdots + w_{s-2} l^{s-2} + z_{n-1} l^{s-1}\big]\end{aligned}$$

with $w_i, \ldots, w_{s-2}$ and $\{z_0, \ldots, z_{n-1}\}$ in $\mathbb{F}_l$ and such that $w_i \equiv 0 \pmod{l}$ implies $w_j \equiv 0 \pmod{l}$ for all $i \leq j \leq s-2$. Let $R = [w_i, w_{i+1}, \ldots, w_{s-2}]$. Let $e = d^{-1} \pmod{l^s}$. The condition $X^d \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$ implies that $X^{de} \pmod{\langle f, p\rangle} \notin \mathbb{F}_p$ and so there exist a pair of distinct elements in $\{z_0, \ldots, z_{n-1}\}$. There are two cases to consider:

- $R = 0$,
- one element in R is nonzero.

Algorithm 10 reduces the second case to the first case in the following way. Consider the constant term of v. Let $z = (-1)^n v(0)$, and $v_1, \ldots, v_n$ be the n elements of the tuple $\operatorname{ind}(v)$. Let $t = \sum_{i=0}^{n} v_i$ then $z = \omega^t$ for some l^s primitive root of unity ω and

$$t = n\big(w_i l^i + w_{i+1} l^{i+1} + w_{i+2} l^{i+2} + \cdots + w_{s-2} l^{s-2}\big) + r l^{s-1}$$

for some positive integer r with $0 \leq r < l$. Let $b = n^{-1} \pmod{l^s}$ then

$$tb = \big(w_i l^i + w_{i+1} l^{i+1} + w_{i+2} l^{i+2} + \cdots + w_{s-2} l^{s-2}\big) + br l^{s-1}.$$

We redefine v as $v = v(\omega^{tb}) = v(yX)$ where y is given in the algorithm. Then

$$\operatorname{ind}(v) = [z_0 l^{s-1}, z_1 l^{s-1}, \ldots, z_{n-1} l^{s-1}].$$

Let $Z = \{bz_0, bz_1, \ldots, bz_{n-1}\}$. At this point there are two cases to consider:

- $0 \in Z$,
- all elements of Z are nonzero.

In the first case, if an element of Z is equal to zero then $X - 1$ is a factor of v. Otherwise

$$bz_0, bz_1, \ldots, bz_{n-1} \in \mathbb{F}_l^*.$$

All the cosets of $\mathbb{F}_l^*$ must have $d \mid l-1$ elements but $\gcd(n, l-1) = 1$ so Z is not a coset of this group. As Z is not a multiplicative coset, we can choose a positive integer $i < l$ so that $0 < \deg(\gcd(v, v^{(i)})) < n$. Let h be this common factor. We need to manipulate h to recover a factor of the original polynomial v so we let $h = h(y^{-1}X)$.

The rest of the algorithm works upwards to recover a factor of $g = M(\alpha)$ from the factor h of v. Let $h' = h(X^l)$ then for any root b of h' there exists a unique root c of h' so that $\text{ind}(b)[l^{s-2}] = \text{ind}(c)[l^{s-1}]$. We have that A_{z-2} is the minimal polynomial of an lth root of β and so $\deg(\gcd(A_{z-2}, h')) = \deg(h)$. Continuing through each iteration of the loop a $\deg(h)$ factor of A_{z-i} is lifted to a $\deg(h)$ factor of A_{z-i+1} until a factor of g is found. □

Given a polynomial f over a finite field $\mathbb{F}_q$ such that $f = f_1^{e_1} \cdots f_k^{e_k}$ for irreducible polynomials $f_1, \ldots, f_k$ let $\text{rad}(f) = f_1 \cdots f_k$.

Theorem 4.17. *Let $\mathbb{F}_q$ be a finite field, f be a squarefree polynomial in L_n over $\mathbb{F}_q$ for $n > 1$, l be a prime dividing $q-1$, $s = \nu_l(q-1)$,*

$$\begin{aligned} d_1 &= l^s, & e_1 &= d_1^{-1} \pmod{(q-1)/l^s}, \\ d_2 &= (q-1)/l^s, & e_2 &= d_2^{-1} \pmod{l^s}, \end{aligned}$$

and $g = f^{d_1 e_1}$, $h = f^{d_2 e_2}$. Then $\deg(\text{rad}(g)) > 1$ or $\deg(\text{rad}(h)) > 1$ and if t is a nontrivial factor of $\text{rad}(g)$ (respectively $\text{rad}(h)$) then $\gcd(t \otimes h, f)$ (respectively $\gcd(t \otimes g, f)$) is a nontrivial factor of f.

Proof. Let $f = \prod_{i=1}^{n} (X - \omega_i)$ then

$$\text{rad}(g) = \prod_{i=1}^{m_1} (X - \gamma_i) \qquad \text{rad}(h) = \prod_{i=1}^{m_2} (X - \beta_i)$$

for $\gamma_1, \ldots, \gamma_{m_1}$ that have order dividing d_1 and $\beta_1, \ldots, \beta_{m_2}$ that have order dividing d_2. Note that d_1 and d_2 are coprime and for each root γ of g (respectively h) there exists a root β of h (respectively g) so that $\gamma\beta$ is a root of f. Hence f divides $g \otimes h$.

If $\deg(\text{rad}(g)) = 1$ and $\deg(\text{rad}(h)) = 1$ then $g \otimes h$ has only one root. As the degree of f is larger than one and f is squarefree we cannot have $\deg(\text{rad}(g)) = 1$ and $\deg(\text{rad}(h)) = 1$.

Without loss of generality let t be a nontrivial factor of $\text{rad}(g)$. Let γ be a root of $\text{rad}(g)$ that is not a root of t. Then there is a root ω of f such that $\omega = \gamma\beta$ for some root β of $\text{rad}(h)$ and this ω is not a root of $t \otimes h$. Hence $\gcd(t \otimes h, f)$ is a nontrivial factor of f. □

We now prove Theorem 4.15.

Proof. The algorithm FIND_PRIME constructs a prime l so that with $s = \nu_l(p-1)$ we have $X^{(p-1)/l^s} \notin \mathbb{F}_p$. Let ω be an l^s primitive root of unity and γ be a $(p-1)/l^s$ primitive root of unity. Then the polynomial f can be written

$$f = \prod_{i=0}^{n-1} (X - \omega^{e_i}\gamma^{j_i}).$$

The polynomial t in Algorithm 11 is

$$t = \prod_{i=0}^{n-1} (X - \gamma^{j_i}).$$

The polynomial v in Algorithm 10 is

$$v = \prod_{i=0}^{n-1} (X - \omega^{e_i})$$

and this polynomial is squarefree. The polynomial h returned by Algorithm 10 is a nontrivial factor of v by Lemma 4.17. Hence a is a nontrivial factor of f by Lemma 4.17. □

4.4 Bounded Degree Polynomials

The rest of this chapter regards the factorization of polynomials of bounded degree. Let $\mathbb{F}_q$ be a finite field, c be a positive real number, and $B = \{f \in L_n : \deg(f) < c\}$ the class of polynomials of bounded degree. In studying the FPFF it is interesting to consider the special case of factoring polynomials of this type. The following result was given in [30].

Theorem 4.18. *Let $\mathbb{F}_q$ be a finite field, and f be a polynomial of bounded degree over $\mathbb{F}_q$. Assuming GRH, we can find all roots of f in polynomial time.*

Although finding a deterministic polynomial time algorithm that does not rely on GRH appears to be a very hard problem in general, the problem of giving deterministic polynomial time algorithms for factoring polynomials of bounded degree may be easier. Theorem 4.21 gives a new result in this direction. This result should be compared to the fastest general algorithm for factoring polynomials over $\mathbb{F}_p$ which was given in Chapter 2.

The next lemma is from [35].

Lemma 4.19. *Let $\mathbb{F}_q$ be a finite field. Then a primitive root in $\mathbb{F}_q$ can be found in $O(q^{1/4+o(1)})$ operations.*

Corollary 4.20. *Let $\mathbb{F}_q$ be a finite field and l a prime dividing $q-1$. Then an lth nonresidue can be found in $O\big(q^{1/4+o(1)}\big)$ operations.*

Proof. By Lemma 4.19 we can construct an lth primitive root ζ in $\mathbb{F}_q$. Hence ζ is an rth nonresidue for each prime r dividing $q-1$. In particular, ζ is an lth nonresidue. □

Theorem 4.21. *Let $\mathbb{F}_p$ be a finite field, and f be a polynomial of bounded degree over $\mathbb{F}_q$. Then f can be factored deterministically in* $\text{poly}\left(p^{1/4+o(1)}\right)$ *operations.*

Proof. We proceed by induction on the least prime l dividing the degree n of the polynomial to be factored. Suppose we have $l = 2$. Then by Corollary 4.20 we can construct a quadratic nonresidue in $\left(p^{1/4+o(1)}\right)$ operations. Given a quadratic nonresidue we can factor f deterministically in $\text{poly}(\log(p))$ operations by Theorem 3.7.

Now let $l_1, \ldots, l_t$ be the first t prime numbers and suppose we can factor any polynomial of bounded degree m so that at least one of $l_1, \ldots, l_{t-1}$ divides m. We wish to show that this is sufficient to construct an algorithm that factors any polynomial g of degree n so that the least prime factor of n is l_t.

Let h be an irreducible factor of Φ_{l_t}. If we can construct a field extension of the form $\mathbb{F}_q = \mathbb{F}_p[X]/(h)$ then by Theorem 4.20 we can construct an lth nonresidue in $\mathbb{F}_q$ and then by Theorem 5.14 a factor of f can be constructed deterministically in $\text{poly}(\log(p))$ operations. To prove the inductive step we need only show it is possible to factor Φ_{l_t}.

The degree of Φ_{l_t} is $l_t - 1$ so by Theorem 2.31, finding an irreducible factor h of Φ_{l_t} in $\mathbb{F}_p$ reduces to finding a root of a polynomial f of degree at most $\deg(\Phi_{l_t}) = l_t - 1$ in $\mathbb{F}_p$. In particular, the degree is divisible by a prime less than l_t. So by the inductive assumption we can factor $f = f_1 f_2$ where without loss of generality $\deg(f_1) < (l-1)/2$. As the degree of f_1 is less than l_t it factors as a product of primes less than l_t. Hence we can factor f_1. We repeat these factorizations until a root of f is found. As the degree of the polynomial to be factored is halved at each step at most $\log(l_t - 1)$ factorizations are needed but since l_t is bounded this is simply a constant number of operations. So the total number of operations for all the factorizations is $\text{poly}\left(p^{1/4+o(1)}\right)$ by the recursive assumption.

Having found a root of f we proceed with Theorem 2.31 to construct an irreducible factor of Φ_{l_t} which takes $\text{poly}(\log(p))$ operations. We then use Theorem 7 to factor g deterministically which takes $\text{poly}(\log(p))$ operations. The number of operations for the full algorithm is then $\text{poly}\left(p^{1/4+o(1)}\right)$.

□

5 Results conditional on GRH

5.1 Factoring even degree polynomials under ERH

The main result of this section is Theorem 5.14. This result was originally proved in [29] and was proved by an alternate method in [15]. The goal of this chapter is to give a proof of this theorem that requires less background knowledge than the previous proofs.

We work up to Theorem 5.14 in steps. In Theorem 5.8 we prove that even degree polynomials with a certain property can be factored in deterministic polynomial time under GRH. In Theorem 5.8 we show that *all* even degree polynomials can be factored in deterministic polynomial time. These results are generalized in Theorem 5.14.

The high level idea for all these results is similar to that of Theorem 4.12. We do a composed operation on the polynomial f to get a polynomial g, use some structure of the resulting polynomial to do a factorization of g, and then "reverse" the composed operation and apply Algorithm 7 to attempt to recover a factor of the original polynomial.

In the case of Theorem 5.8 the composed operations is $\ominus$. This operation is reversed by $\oplus$. We begin by constructing $g = X^{-n}(f \ominus f)$. The choice of $\ominus$ as the composed operation is motivated by the following observation from Chapter 2. If r_1, r_2 are elements of $\mathbb{F}_p$ and $\text{ind}(r_1 - r_2) \equiv k \pmod{2^s}$ then $\text{ind}(r_2 - r_1) \not\equiv k \pmod{2^s}$. Hence g can be factored by Algorithm 6.

Factoring g is not sufficient to be able to construct a factor of f. We call factors of g that do not permit a factor of f to be recovered $\ominus$-*balanced*. We give several different types of $\ominus$-balance. The simplest, used in Algorithm 12 is Definition 5.1. The definition of $\ominus$-balanced polynomials mirrors Definition 4.8.

Definition 5.1. Let $\mathbb{F}_q$ be a finite field, let $f \in L_n$, let R be the set of roots of f, and let $t_1 = X^{-n}(f \ominus f)$. A factor h of t_1 is $\ominus$-*balanced* with respect to f if for each root $a \in R$

$$\#\{b \in R : a \neq b \text{ and } h(a-b) = 0\} = \frac{n-1}{2}.$$

As an example of Definition 5.1 if $f = (X-(r_1-r_2))(X-(r_2-r_3))(X-(r_3-r_1))$ for some r_1, r_2, r_3 in a finite field $\mathbb{F}_q$ then f is balanced. We note that every even degree polynomial is not balanced. Indeed, Algorithm 15 can factor any even degree polynomial deterministically in polynomial time under GRH.

Next we define l-index decompositions and (l,k)-index partitions which are used throughout this chapter. We give an example to motivate Definition 5.2. Recall the example we gave in Chapter 2 of factoring a polynomial using Algorithm 6. In this example we took p to be a prime so that $s = \nu_2(p-1) = 4$, and z be a quadratic nonresidue modulo p. We let a and b be distinct elements of $\mathbb{F}_p$ of odd order, and let

$$f = (X - az)(X - bz)(X - az^3)(X - bz^3)(X - az^{13})(X - bz^{13}).$$

We found the factorization $f = g_1 g_2 h_2$ with

$$g_1 = (X - az)(X - bz) \qquad g_2 = (X - az^{13})(X - bz^{13}) \qquad h_2 = (X - az^3)(X - bz^3).$$

This is as far as we could take factorization f using Algorithm 6. In particular we observed that if a and b are arbitrary roots of distinct elements of the set $\{g_1, g_2, h_2\}$ then $\text{ind}(a) \not\equiv \text{ind}(b) \pmod{2^s}$.

Definition 5.2. Let $\mathbb{F}_q$ be a finite field, l a prime diving $q-1$, $s = \nu_l(q-1)$, and $f \in L_n$. Then a set of polynomials T is an *l-index decomposition* of f if $f = \prod_{t \in T} t$ and for all $t \in T$ and for all pairs $z_1, z_2 \in \mathbb{F}_q$ such that $(X - z_1) \mid t$ and $(X - z_2) \mid t$ we have $\text{ind}(X - z_1) \equiv \text{ind}(X - z_2) \pmod{l^s}$.

The set $\{g_1, g_2, h_2\}$ given before the example is an 2-index decomposition of the polynomial f.

Algorithm 13 shows how to construct an l-index decomposition. Algorithm 12 is an auxiliary algorithm for Algorithm 13 that takes a prime l and a list of polynomials and calls Algorithm 6 on each polynomial in the tuple. Algorithm 12 makes use of the following algorithms:

- IND_FAC - Algorithm 6.

Algorithm 12: SPLIT

Input : $\mathbb{F}_q, l, k, A, \zeta$
$\mathbb{F}_q$ a finite field
l a prime dividing $q-1$ such that $s = \nu_l(q-1)$
k a nonnegative integer so that $k < s$
A a set of m polynomials $g_1, \ldots, g_m$ such that g_i is a product of linear factors and for every $g_i, g_j \in A$ we have $\text{ind}(g_i)[l^t] \neq \text{ind}(g_j)[l^t]$ for $k < t \leq s-1$
ζ an lth nonresidue
Output : A_2 a set of m_2 polynomials $h_1, \ldots, h_{m_2}$ such that h_i is a product of linear factors and for every $h_i, h_j \in A_2$ we have $\text{ind}(h_i)[l^t] \neq \text{ind}(h_j)[l^t]$ for $k+1 < t \leq s-1$

```
Initialize A_2 as an empty list
for a ∈ A do
    a_1 = IND_FAC(F_q, l, ζ, a)
    a_2 = a/a_1
    if 0 < deg(a_1) and deg(a_1) < deg(a) then
        Add a_1 and a_2 to A_2
    end
    else
        Add a to A_2
    end
end
Return A_2
```

Algorithm 13 constructs an l-index decomposition by repeatedly calling Algorithm 12. Algorithm 13 makes use of the following algorithms:

- IS_SQF
- SQRFREE_DECOMP - Algorithm 1
- SPLIT - Algorithm 12

Algorithm 13: DECOMP

Input : $\mathbb{F}_q, l, f$
$\mathbb{F}_q$ a finite field
l a prime dividing $q-1$
$f \in L_n$ for some positive integer n
Output : l-index decomposition of f

Construct an lth nonresidue ζ in $\mathbb{F}_q$ using Theorem 3.6
/* The algorithm is simpler when the polynomial is squarefree so we split the algorithm into two cases */
if *IS_SQF*$(f) = true$ **then**
  $s = \nu_l(q-1)$
  $D = [f]$
  for $i = 0$ **to** $s-1$ **do**
    /* D is a tuple of k elements $d_1, \ldots, d_k$ */
    Initialize B as an empty tuple
    for $j = 1$ **to** k **do**
      $B = B \cup \text{SPLIT}(\mathbb{F}_q, l, d_j, \zeta)$
    end
    $D = B$
  end
  Return A
end
else
  /* $A = \{a_1, \ldots, a_m\}$ is a squarefree decomposition of $f = g_1^{e_1} \cdots g_k^{e_k}$ with $e_1 < e_2 < \cdots < e_k$ */
  /* a_i is a pair $[g_k, e_k]$ */
  Let $a_{i,0}$ be the first element of a_i
  $A = \text{SQRFREE\_DECOMP}(f, \mathbb{F}_q)$
  $m = \#A$
  /* Initialize d as an empty tuple $d = [d_1, \ldots, d_m]$ */
  /* d_i will hold a tuple of polynomials */
  for i **to** m **do**
    /* Recursive call of DECOMP for each element in the squarefree decomposition */
    Let $d_{i,j}$ be the jth element of d_i
    /* The recursive call enters the squarefree case */
    $d_i = \text{DECOMP}(\mathbb{F}_q, l, a_{i,0}, \zeta)$
    for j **from** 1 **to** $\#d_i$ **do**
      $d_{i,j} = (d_{i,j})^i$
    end
  end
  Return d
end

For example, let p be a prime, f be a squarefree polynomial in $\mathbb{F}_p[X]$ such that $f = (X - r_1)(X + r_1)(X - r_2)(X + r_2)$ and let $s = \nu_2(p-1) = 1$ then we have $\text{ind}(-r_1)[2^{s-1}] \in \{0, 1\}$ and $\text{ind}(-r_1)[2^{s-1}] \neq \text{ind}(r_1)[2^{s-1}]$. We apply Algorithm 13 to f which gives $D = [f]$. After SPLIT is applied to A we have $D = [(X-r_1)(X-r_2), (X+r_1)(X+r_2)]$. The algorithm returns this D. If instead we had $s = \nu_2(p-1) = 2$

then SPLIT would be applied to each of the polynomials $(X - r_1)(X - r_2), (X + r_1)(X + r_2)$.

Theorem 5.3. *Let $\mathbb{F}_q$ be a finite field, l a prime dividing $q-1$. Then Algorithm 13 constructs an l-index decomposition in deterministic* $\text{poly}(\log(q), n, l)$ *operations under GRH.*

Proof. First we consider the squarefree case. The number of elements in A is never more than n. Each recursive call to Algorithm 12 consists of at most n calls to Algorithm 6 plus some simple algebraic manipulations. By Theorem 2.35 the call to Algorithm 6 can be done within the given number of operations. By Theorem 2.29 the other operations can be done within the given number of operations. The non-squarefree case consists of iterating over a tuple of at most n elements and making a call to DECOMP with a squarefree polynomial for each element in the tuple and raising an element to a power. This exponentiation can be done in the time bound by Theorem 2.29. □

Definition 5.4. Let $\mathbb{F}_q$ be a finite field, f a polynomial in L_n, l be a prime dividing $q-1$, and k be a nonnegative integer less than $\nu_l(q-1)$. A set of l polynomials $M = \{m_0, \ldots, m_{l-1}\}$ over $\mathbb{F}_q$ is an (l,k)-*index partition* if $f = \prod_{j=0}^{l-1} m_j$ and $\text{ind}(m_j)[l^k] = j$.

We give an example of constructing a (2,2) index partion of a polynomial f. This example follows the example used for Definition 5.2. We take p a prime, $s = v_2(p-1) = 4$, z a quadratic nonresidue, and a, b elements of odd order, and

$$f = (X - az)(X - bz)(X - az^3)(X - bz^3)(X - az^{13})(X - bz^{13}).$$

Let $l = 2$, $k = 3$, and $A = [f]$. As $k = s - 1$ there are no integers t so that $k < t \leq s - 1$ so the condition on A is met. We construct the quadratic nonresidue ζ modulo p. The roots of the polynomial f are roots $az, -bz, -az^3, -bz^3, -az^{13}, -bz^{13}$. In Chapter 2 we saw that the base 2 representations of the indexes are $[1, 1, 11, 11, 1101, 1101]$. Hence the set of polynomials

$$\{(X - az)(X - bz)(X - az^3)(X - bz^3), (X - az^{13})(X - bz^{13})\}$$

is a $(2,2)$-index partition of the polynomial f.

Algorithm 14 gives the pseudocode for partitioning a polynomial based on its 2-index. The algorithm for general $l \mid q - 1$ is similar but more involved.

Algorithm 14: PARTITION

Input : $\mathbb{F}_q, f, M$
  $\mathbb{F}_q$ a finite field
  M is a 2-index decomposition of f
Output : A $(2, s-1)$-index partition of f
$n = \#M$
$L_0 = 1$
$L_1 = 1$
$s = \nu_2(q-1)$
$R = \mathbb{F}_q[X]/(f)$
for i **from** *1* **to** n **do**
  $f = M[i]$
  `/* t is an element in the ring R */`
  $t = X^{(q-1)/2^s}$
  if $\deg(t, X) > 0$ **then**
    Return FAIL
  end
  if $\mathrm{ind}(t)[2^{s-1}] = 0$ **then**
    $L_0 = f \cdot L_0$
  end
  else if $\mathrm{ind}(t)[2^{s-1}] = 1$ **then**
    $L_1 = f \cdot L_1$
  end
  else
    Return FAIL;
  end
end
Return $[L_0, L_1]$

Theorem 5.5. *Let $\mathbb{F}_q$ be a finite field, f a polynomial in L_n, l be a prime dividing $q-1$, and k be a nonnegative integer less than $\nu_l(q-1)$. Then there exists an algorithm, of which Algorithm 14 is a special case, that constructs an (l,k)-index partition of f in* $\mathrm{poly}(\log(q), n, l)$ *operations.*

The next lemma is an application of Algorithm 14. A polynomial $f \in L_n$ such that $f = g(X^m)$ for some polynomial $g \in L_{n/m}$ and positive integer m is called an *m-cycle polynomial*.

Theorem 5.6. *Let $f \in L_n$ be a m-cycle polynomial over $\mathbb{F}_q$. If an mth nonresidue in $\mathbb{F}_q$ is given then f can be factored as $f_1 \cdots f_m$ such that the polynomials f_i have the same degree and are pairwise relatively prime.*

Proof. Let g be the polynomial such that $f = g(X^m)$ and let $s = \nu_m(q-1)$. If ζ is a root of g then all the roots of the polynomial $X^m - \zeta$ are roots of f. If α is a root of $X^m - \zeta$ and ω is an primitive mth root of unity then $\zeta, \zeta\omega, \ldots, \zeta\omega^{m-1}$ are all the roots of $X^m - \zeta$. The indexes of this set are $\mathrm{ind}(\zeta\omega^i) = \mathrm{ind}(\zeta) + \mathrm{ind}(\omega^i) = \mathrm{ind}(\zeta) + i \cdot \mathrm{ind}(\omega)$ hence $\mathrm{ind}(\zeta\omega^i) \equiv \mathrm{ind}(\zeta) + im^{s-1} \pmod{m^s}$. This equality shows that the $(m, s-1)$ index partition consists of the following r polynomials $t_0, \ldots, t_{r-1}$ of

degree n/m with

$$t_i = \prod_{\text{ind}(r)[m^{s-1}]=i} (X - r).$$

□

This theorem is given in [29] as Proposition 2.3. (This proposition calls f for which $f = g(X^r)$ *r-good* polynomials. Although Proposition 2.3 regards *r-good* polynomials with an additional property, all r-good polynomials turn out to have this property).

We now give an algorithm for factoring polynomials of even degree. Let f be a polynomial of degree n in L_n. The algorithm is based on the following idea. Let r_i, and r_j be roots of f. If $\text{ind}(r_i - r_j) \equiv k \pmod{2^s}$ then as we saw in an example in Chapter 2, $\text{ind}(r_j - r_i) \not\equiv k \pmod{2^s}$. Hence Algorithm 5 produces a factor of $h = X^{-n}(f \ominus f)$. We apply Algorithm 7 to h to recover a factor of the original polynomial f.

Algorithm 15 makes use of the following algorithms:

- IS_SQF
- DECOMP - Algorithm 13
- PARTITION - Algorithm 14
- BASE - Given a polynomial f with factorization $f_1^{e_1} \cdots f_k^{e_k}$ over $\mathbb{F}_q$ $\text{BASE}(f) = f_1 \cdots f_k$
- NON_FLAT - Algorithm 7

Algorithm 15: EVEN_DEG

Input : $\mathbb{F}_q, f$
  $\mathbb{F}_q$ a finite field
  f in L_n, f squarefree, and n even
$s = \nu_2(q-1)$
$h = X^{-n}(f \ominus f)$
$d = \text{DECOMP}(\mathbb{F}_q, 2, f)$
`/* t is a pair of polynomials` t_1, t_2 `of degree` $n(n-1)/2$ `*/`
$t = \text{PARTITION}(\mathbb{F}_q, d)$
$T = \text{BASE}(t)$
$g = T \oplus f$
Return $\text{NON\_FLAT}(\mathbb{F}_q, f, g)$

As in Chapter 4, it may be useful to interpret the steps of Algorithm 15 in terms of graphs. In Definition 2.39 we defined arc polynomials. Given a polynomial f on n vertices $r_1, \ldots, r_n$ then the arc polynomial corresponding to $h = X^{-n}(f \ominus f)$ is the complete directed graph.

Now suppose, for example, we are given a prime $p \equiv 3 \pmod 4$. We apply Algorithm 5 to this polynomial which gives a factor t_0 of h with $\deg(t_0) = n(n-1)/2$ of h because for any pair of roots r_1, r_2 exactly one of $\{r_1 - r_2, r_2 - r_1\}$ is a quadratic nonresidue. In the corresponding arc polynomial exactly one of the arcs $\{[r_1, r_2], [r_2, r_1]\}$ appears. Hence the graph represented by the arc polynomial t_0

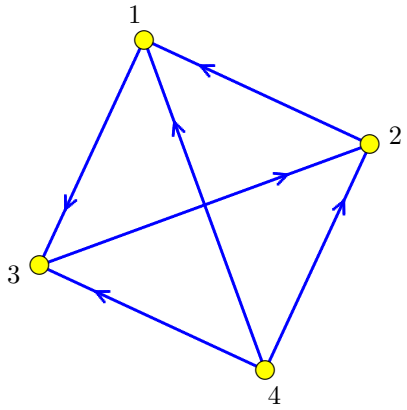

Figure 5: A tournament on four vertices

is a tournament.

For example, let $p = 103$ and let $\chi(a) = a^{(p-1)/2}$ for $a \in \mathbb{F}_p$. Let $r_1 = 5, r_2 = 17, r_3 = 23, r_4 = 101$, and

$$\begin{aligned}
v_1 &= \chi(r_1 - r_2) = 1 & v_2 &= \chi(r_2 - r_1) = -1,\\
v_3 &= \chi(r_1 - r_3) = -1 & v_4 &= \chi(r_3 - r_1) = 1,\\
v_5 &= \chi(r_1 - r_4) = 1 & v_6 &= \chi(r_4 - r_1) = -1,\\
v_7 &= \chi(r_2 - r_3) = 1 & v_8 &= \chi(r_3 - r_2) = -1,\\
v_9 &= \chi(r_2 - r_4) = 1 & v_{10} &= \chi(r_4 - r_2) = -1,\\
v_{11} &= \chi(r_3 - r_4) = 1 & v_{12} &= \chi(r_4 - r_3) = -1.
\end{aligned}$$

As a result, the polynomial t_0 that we recover is

$$\big(X - (r_1 - r_2)\big)\big(X - (r_3 - r_1)\big)\big(X - (r_1 - r_4)\big)\big(X - (r_2 - r_3)\big)\big(X - (r_2 - r_4)\big)\big(X - (r_3 - r_4)\big).$$

The corresponding tournament is shown in Figure 5.1. In this figure the vertices $1, 2, 3, 4$ correspond to the roots r_1, r_2, r_3, r_4. The edges correspond to the arc polynomial t_0.

Definition 5.7. Let $\mathbb{F}_q$ be a finite field, $s = \nu_2(q-1)$, $f \in L_n$ over $\mathbb{F}_q$ with roots $R = \{r_1, \ldots, r_n\}$, and $h = X^{-n}(f \ominus f)$ with roots H. The 2-*score* $S(r)$ of $r \in R$ is the number of elements $r_i - r_j$ of H such that $\mathrm{ind}(r_i - r_j)[2^{s-1}] = 0$. The *score polynomial* L is defined $L = \prod_{i=1}^{n} (X - r_i)^{S(r_i)}$.

Theorem 5.8. *Let $\mathbb{F}_q$ be a finite field, and f be a polynomial in L_n of even degree. Under ERH Algorithm 15 factors f deterministically in* $\mathrm{poly}(n, \log(q))$ *$\mathbb{F}_p$-operations.*

Proof. If $n > p$ we can factor f in deterministic polynomial time using Theorem 3.1. If $n < p$ construct the polynomial $h = X^{-n}(f \ominus f)$. The polynomial h is a 2-cycle polynomial and so if R denotes the set of roots of f then half the elements $r \in R$ are such that $\text{ind}(r)[2^{s-1}] = 0$ and the other half $\text{ind}(r)[2^{s-1}] = 1$. Let T be the factor of h whose roots are $\{a_i : h(a_i) = 0 \text{ and } \text{ind}(a_i)[2^{s-1}] = 1\}$. Due to the assumption of ERH we can construct a quadratic nonresidue in deterministic polynomial time. Given this quadratic nonresidue the polynomial T can be constructed by Algorithm 14.

Let $g = T \oplus f$. The score polynomial L is a divisor of g. The roots of h have the form $x - y$ for some roots x, y of f which are not equal. So the roots of g have the form $x - y + z$. Clearly if $y = z$ then $x - y + z = x$. We want to show that none of the roots of g/L are also roots of f. This is equivalent to showing that the multiplicity of the factor $(X - r)$ of g is equal to the score of r. Let $(X - r)$ be a factor of g with multiplicity k. Then there are k solutions $x, y, z \in R$ such that $x - y$ is a root of t to

$$r = x - y + z$$

$$r - z = x - y.$$

Hence the multiplicity of $(X - r)$ is the number of pairs of distinct elements $x, y \in R$ such that $x - y$ is a root of t and is equal to $r - z$ for some z in R. Hence the multiplicity of $(X - r)$ is equal to the score of r.

The score polynomial L can be computed as $L = \gcd(f^{n-1}, g)$. As n is even, n does not divide $n(n-1)/2$ so there are at least two distinct scores. Hence L is not flat with respect to f so by applying Algorithm 7 we can recover a factor. □

5.2 Factoring generic polynomials under GRH

We now discuss how to generalize Theorem 5.8 result to arbitrary polynomials. The following lemma is needed for the proof of Theorem 5.14.

Lemma 5.9. *Let $\mathbb{F}_q$ be a finite field, $f \in L_n$, ω be a primitive lth root of unity, and let $g_i = \omega^i f$ for $i = 0, \ldots, l-1$*

$$\prod_{i=0}^{l-1} (X - g_i) = X^l - f^l.$$

Proof. As we saw in Chapter 2 $X^d - 1$ is a product of all the elements of order d in $\mathbb{F}_q$. Hence the roots of $f^{-l}(X^l - f^l) = (X/f)^l - 1$ is the set of dth roots of unity in $\mathbb{F}_q$. The result follows. □

The next definition generalizes Definition 5.1.

Definition 5.10. Let l be a prime and $\mathbb{F}_q$ be a finite field so that $q \equiv 1 \pmod{l}$. Let $f = (X - r_1)\cdots(X - r_n)$ for $r_i \in \mathbb{F}_q$. A polynomial h of degree

$$\frac{n(n-1)(n-2)\cdots(n-l+1)}{l}$$

is $(\ominus, l)$-*balanced* with respect to f if for each root λ of f

$$\#\{r_{k_0}, \ldots, r_{k_{l-2}} \in R : h(r_{k_0} + \omega r_{k_1} + \ldots + \omega^{l-2} r_{k_{l-2}} + \omega^{l-1}\lambda) = 0\} = \frac{(n-1)(n-2)\cdots(n-l+1)}{l}.$$

If l is a prime factor of n then h is not $(\ominus, l)$-balanced with respect to f.

We now discuss how to generalize Theorem 5.8 result to arbitrary polynomials. Let

$$f = \prod_{i=1}^{n} (X - r_i)$$

be a degree n polynomial over $\mathbb{F}_p$. Let l be the least prime factor of n. Under GRH by Theorem 3.6 we can construct an extension $\mathbb{F}_q$ of $\mathbb{F}_p$ that contains primitive lth primitive roots of unity. Let $Y_1, \ldots, Y_n$ be indeterminants and G be the set of permutations of subsets of l elements of $[Y_1, \ldots, Y_n]$. Every element in G has the form $[Y_{i_1}, \ldots, Y_{i_l}]$ such that $1 \leq i_k \leq n$ and all the i_k are distinct, and let $R = \mathbb{F}_q[Y_1, \ldots, Y_n]$. We define the linear map $\sigma : R \to \mathbb{F}_q$

$$\sigma(g) = \sigma([Y_{i_1}, \ldots, Y_{i_r}]) = [r_{i_1}, \ldots, r_{i_r}].$$

for any g in G.

Let ω be an lth primitive root of unity in $\mathbb{F}_q$. We define a linear polynomial in R on the elements of G

$$L(g) = L([Y_{i_0}, \ldots, Y_{i_{l-1}}]) = \sum_{k=0}^{l-1} \omega^k Y_{i_k}.$$

for any $g \in G$. When $l = n$ this polynomial is sometimes referred to as the *Lagrange resolvent*. It appears in Galois theory, for example in the derivation of Cardano's formula for expressing the roots of a cubic polynomial in terms of radicals. For general l we will call it the *resolvent polynomial* of g.

Now we define a polynomial over $\mathbb{F}_q[X]$ which has $\sigma(L(g))$ for each $g \in G$ as its roots. Let

$$t_G = \prod_{g \in G} \Big(X - \sigma\big(L(g)\big)\Big).$$

Let S be the a subset of G of $\#G/n$ elements such that no elements of S are in the same r-cycle. By

Lemma 5.9 we can regroup the terms of $t(G)$ as

$$t_G = \prod_{s \in S} \Big(X^l - \sigma\big(L(s)\big)^l \Big). \tag{2}$$

Hence t_G is a l-cycle polynomial.

Our goal is to construct the polynomial t_G. We have already seen how to construct t_G in the case $l = 2$ using the composed difference of $f \ominus f$. In order to construct this polynomial we continue use the composed sum of a set of polynomial whose roots are related to f by a multiplicative factor. However, using the composed sum in this way generates some factors that are not factors of t_G. These factors must then be divided out. For example, if $\mathbb{F}_q$ is a finite field and we are given the three polynomials

$$f_0 = (X - \lambda_0 r_1)(X - \lambda_0 r_2)(X - \lambda_0 r_3)$$
$$f_1 = (X - \lambda_1 r_1)(X - \lambda_1 r_2)(X - \lambda_1 r_3)$$
$$f_2 = (X - \lambda_2 r_1)(X - \lambda_2 r_2)(X - \lambda_2 r_3)$$

with $r_1, \ldots, r_n \in \mathbb{F}_q$ and $a, b \in \mathbb{F}_q$ and we let $g = f_1 \oplus f_2 \oplus f_3$ and

$$v_1 = \lambda_0 r_1 + \lambda_1 r_2 + \lambda_2 r_3$$
$$v_2 = \lambda_0 r_1 + \lambda_1 r_3 + \lambda_2 r_2$$
$$v_3 = \lambda_0 r_2 + \lambda_1 r_1 + \lambda_2 r_3$$
$$v_4 = \lambda_0 r_2 + \lambda_1 r_3 + \lambda_2 r_1$$
$$v_5 = \lambda_0 r_3 + \lambda_1 r_1 + \lambda_2 r_2$$
$$v_6 = \lambda_0 r_3 + \lambda_1 r_2 + \lambda_2 r_1$$

$$h = (X - v_1)(X - v_2)(X - v_3)(X - v_4)(X - v_5)(X - v_6)$$

then $h \mid g$. If we take $\lambda_i = \omega^i$ with ω a primitive 3rd root of unity then $h = t_G$. Hence we can construct t_G by first finding g and then removing the 21 roots of g that are not roots of h. We now define the type of factors that we must remove from g.

Definition 5.11. Let $\mathbb{F}_q$ be a finite field and f be a polynomial with roots $\{r_1, \ldots, r_n\}$ over $\mathbb{F}_q$ and let l be a prime dividing $q - 1$ and dividing l. Then h is a *i-common pair* polynomial for $0 \leq i \leq l$ if its roots have the form $r_{j_0} + \omega r_{j_1} + \cdots + \omega^{i-1} r_{j_{i-1}} + \omega^i r_{j_k}$ where $r_{j_0}, r_{j_1}, r_{j_{i-1}}, r_{j_k}$ are all distinct and $r_{j_k} = r$ for some $r \in \{r_{j_0}, r_{j_1}, \ldots, r_{j_{i-1}}\}$.

For example let $\mathbb{F}_q$ be a finite field with 3rd primitive roots of unity ω. Then

$$\begin{aligned} f =&(X-(r_1+\omega r_2+\omega^2 r_2))(X-(r_2+\omega r_1+\omega^2 r_1))(X-(r_2+\omega r_3+\omega^2 r_3)) \\ &\cdot(X-(r_3+\omega r_2+\omega^2 r_2))(X-(r_1+\omega r_3+\omega^2 r_3))(X-(r_3+\omega r_1+\omega^2 r_1)) \end{aligned}$$

is a 2-common pair polynomial. Algorithm 16 takes a polynomial f as input and returns a list of common pair polynomials $h_0, \ldots, h_i$ for some integer i.

Algorithm 16: COM_PAIR

Input : $\mathbb{F}_q, f, \omega, i$
  $\mathbb{F}_q$ a finite field
  f a polynomial in L_n
  ω a root of unity
  i a positive integer
Output : a list of common-pair polynomials
$n = \deg(f)$
/* Initialize P as a list of length i with elements $p_1, \ldots, p_i$ */
for k **from** 1 **to** i **do**
  if $\omega^i + \omega^{k-1} \neq 0$ **then**
    $p_k = f(X/(\omega^i + \omega^{k-1}))$
  end
  else
    $p_k = X^n$
  end
  $p_k = p_k \oplus \sum_{t \neq k} \oplus f(X/\omega^{t-1})$;
end
Return P

Theorem 5.12. *Let p be a prime, $\mathbb{F}_p$ a finite field, f a polynomial in L_n, ω a root of unity, i a positive integer. Algorithm 16 returns a list of all the i-common pair polynomials deterministically in* $\text{poly}(n, \log(p))$ *operations.*

Proof. The elements of p_k are constructed in two steps. First for each pair i, k a polynomial with roots $(\omega^i + \omega^{k-1})r_j$ for each root r_j of f is constructed. Then composed sum of this polynomial is computed with the polynomials $f(X/\omega^{t-1})$ (each root of which is ω^{t-1} multiplied by some root of f). The run time follows by the fact that composed sums $f \oplus g$ can be computed in deterministic polynomial time in $\deg(f)$ and $\deg(g)$. □

Algorithm 17 constructs the polynomial t_G defined earlier. Algorithm 17 makes use of the following algorithms:

- COM_PAIR - Algorithm 16

Algorithm 17: RESOLVENT_POLY

Input : $\mathbb{F}_q, l, f, \omega$
  $\mathbb{F}_q$ a finite field
  l a prime dividing $q-1$
  f a squarefree polynomial in L_n so that $\deg(f) < n^l$
  ω an primitive lth root of unity in $\mathbb{F}_q$
Output : a factor h of f

```
/* Builds up a set of ``partial polynomials''                                    */
```
Find ω an lth primitive root of unity
```
/* H, and h hold a partial sum of the terms of the resolvent polynomial; H holds the current partial
   sum, h holds the next partial sum                                             */
```
$H = 1$
$h = f$
Initialize E as $[E_0, \ldots, E_{l-1}]$
$E = [f(X/\omega^0), \ldots, f(X/\omega^{l-1})]$
for i **from** 1 **to** l **do**
    Initialize S as $[S_0, \ldots, S_i]$
    $S = \text{COM\_PAIR}(f, \omega, i, p)$
    $H = h$
    $h = h \oplus E_i$
    for j **from** 1 **to** $\#S$ **do**
        $g = \gcd(h, S_j)$
        $h = h/g$
    end
end
Return $[H, h]$

Given a polynomial f with roots $r_1, \ldots, r_i$ and the algorithm creates a polynomial h over $\mathbb{F}_q$ with roots of the form $r_1 + \omega r_2 + \cdots + \omega^i r_i$ for each subset $r_1, \ldots, r_i$ of distinct roots of f. The tuple $E = [f(X/\omega^0), \ldots, f(X/\omega^{n-1})]$ holds polynomials with roots that are multiples of ω^i of roots of the polynomial f. Computing the composed sum of h and E_i gives a polynomial with roots of the form $r_1 + \omega r_2 + \cdots + \omega^i r_i + \omega^{i+1}\alpha$ for each subset $r_1, \ldots, r_i$ of roots of f and α a root of f. We define h as this composed sum.

We need to eliminate those factors of h so that α is equal to one of the roots $r_1, \ldots, r_i$. The set of these bad factors is precisely the set of i-common pair polynomials. Dividing these out of h gives a polynomial with roots of the form $r_1 + \omega r_2 + \cdots + \omega^i r_i$ for each subset $r_1, \ldots, r_{i+1}$. Continuing in this way we construct the resolvent polynomial.

Theorem 5.13. *Let $\mathbb{F}_q$ a finite field, l a prime dividing $q-1$, f a squarefree polynomial in L_n so that $\deg(f) < n^l$, ω an primitive lth root of unity in $\mathbb{F}_q$ then Algorithm 17 returns a resolvent polynomial in $\text{poly}(n^l, \log(p))$ operations.*

Proof. We have shown already that Algorithm 17 returns a resolvent polynomial so we only need to bound the number of operations. The degree of the composed sum of two polynomials of degree n has

degree n^2 so in each iteration of the for loop we generate a polynomial h of degree bounded by n^i by computing the composed sum of a polynomial of degree n^{i-1} and n. We have seen above that we can generate the i-common pair polynomials within the time bound. As there are n of these polynomials we can divide h by these polynomials and compute the polynomial g with a *gcd* with h within the time bound as well. □

We are no ready to give an algorithm for factoring arbitrary polynomials deterministically under GRH. In this algorithm we focus on the factorization of polynomials f over $\mathbb{F}_p$. As the factorization of polynomials over $\mathbb{F}_q$ reduces to finding roots of polynomials over $\mathbb{F}_p$ this restriction is not very great. This is restriction is also made in [29] where Theorem 5.12 was originally proved.

Algorithm 18 produces a factor of f in the case that $t_0 \ominus v$ is squarefree. Algorithm 18 makes use of the following algorithms:

- RESOLVENT_POLY - Algorithm 17
- PARTITION - Algorithm 14
- NON_FLAT - Algorithm 7

Algorithm 18: ALGO:FACTOR

Input : p, f
  p a prime
  f a squarefree polynomial in L_n with $n^l < p$
Output : a factor h of f
Let l be the least prime dividing n
Construct the finite field $\mathbb{F}_q$ such that $q \equiv 1 \pmod{l}$
Construct ω a primitive lth root of unity in $\mathbb{F}_q$
$F = \prod_{i=0}^{n-1} (X - \omega^{l-1} r_i)$
/* RESOLVENT_POLY returns a pair of the form (t_G, v) */
$[r, v] = \text{RESOLVENT\_POLY}(\mathbb{F}_q, l, f)$
/* g is the set $\{t_0, \ldots, t_m\}$ */
$g = \text{PARTITION}(\mathbb{F}_q, l, r)$
$z = t_0 \ominus v$
/* g is a nontrivial factor of F */
$b = \text{NON\_FLAT}(z, F, \mathbb{F}_q)$
/* use a substitution to construct a nontrivial factor of f from g */
/* w is a nontrivial factor of f */
$w = b(\omega^{l-1} X)$
Return w

Theorem 5.14. *Let p be a prime, f be a polynomial in L_n over $\mathbb{F}_p$, and l be the least prime divisor of n. Under GRH, Algorithm 18 factors f deterministically in* $\text{poly}(n^l, \log(p))$ $\mathbb{F}_p$ *operations.*

Proof. If $p > n^l$ then we can factor f deterministically using Berlekamp's algorithm. Let the roots of f be $r_0, \ldots, r_{n-1}$. Let $s = \nu_l(q-1)$. Using Theorem 3.6 under GRH we can construct the finite field $\mathbb{F}_q = \mathbb{F}_p[X]/(h)$ where h is an irreducible factor of Φ_l and a primitive lth root of unity $\zeta \in \mathbb{F}_q$. In $\mathbb{F}_q$

construct the polynomials t_G and v using Algorithm 17. The polynomial t_G is an l-cycle polynomial of degree $n(n-1)\cdots(n-l+1)$.

So using Algorithm 5.5 we can construct a $(l, s-1)$ index partition. Let ω be a primitive lth root of unity. Let $F = \prod_{i=0}^{n-1}(X - \omega^{l-1}r_i)$. As t_G is an l-cycle polynomial the index partition consists of l polynomials of degree $n(n-1)\cdots(n-l+1)/l$. Let t_0 be the first polynomial in this partition. The degree of t_0 is not divisible by l so t_0 is not $\ominus$ balanced with respect to F. Hence $b = \text{NON\_FLAT}(z, F, p)$ is a nontrivial factor of F and $w = b(\omega^{l-1}X)$ is a nontrivial factor of f.

For the number of operations, apart from substitutions into polynomials the algorithm consists of calls to RESOLVENT_POLY, PARTITION, NON_FLAT, which we know can be done within the time bound from previous theorems.

□

6 Conclusion

We have studied the deterministic factorization of polynomials over finite fields both unconditionally and under the Riemann Hypothesis. We consider some questions for further investigation.

Question 1. It has been known for some time that polynomials can be factored deterministically in a number of operations proportional to $S(p-1)$ under the Extended Riemann Hypothesis. In Theorem 4.15 we found that polynomials can be factored deterministically and unconditionally in a number of operations proportional to $S(p-1)$ provided that $\gcd(n, \phi(p-1)) = 1$. Can this requirement be weakened?

Theorem 3 in [3] could potentially be useful. This theorem shows that to factor arbitrary polynomials over $\mathbb{F}_p$ in $\text{poly}(\log(p), n, S(p-1))$ operations it is sufficient to find roots of Φ_l for each prime l dividing $p-1$ in $\text{poly}(\log(p), n, S(p-1))$ operations. It seems possible that the condition can at least be improved to

$$\gcd\Big(n, \prod_{\substack{l \mid p-1 \\ \text{l prime}}} (l-1)\Big) = 1.$$

Question 2. The best algorithm for factoring generic polynomials of degree n over the finite field $\mathbb{F}_{p^k}[X]$ uses

$$\log(p)\tilde{O}((nk)^2) + p^{1/2}\log(p)\tilde{O}((nk)^{3/2})$$

$\mathbb{F}_p$ operations. We have considered the class of bounded degree polynomials and found that we can factor polynomials in the class deterministically and unconditionally more quickly than generic polynomials can be factored. Further improvements to the class of bounded degree polynomials may be possible. We restricted attention to the case of unconditionally and deterministically factoring bounded degree polynomials. This class could also be examined under GRH. Are there other classes of polynomials that can be factored deterministically and unconditionally more efficiently than generic polynomials?

Question 3. The class of cyclotomic polynomials is relevant to many results on FPFF in the literature. The ability to factor polynomials in this class unconditionally would remove the requirement of the Generalized Riemann Hypothesis from many theorems. There has been recent progress on the unconditional factorization of cyclotomic polynomials in [17]. However, complete factorization of generic polynomials in this class seems out of reach of current methods. For example, these polynomials do not satisfy the condition $\gcd(n, \phi(p-1)) = 1$ and so cannot be factored using Theorem 4.15. Can the methods of Chapter 4 be extended to give new results on the unconditional factorization of cyclotomic polynomials?

Question 4. The problem of constructing nonresidues is closely related to the factorization of polynomials. For example, there is a polynomial time equivalence between the problem of constructing a

quadratic nonresidue and factoring degree two polynomials. Given an algorithm $\mathcal{A}$ for factoring quadratic polynomials over $\mathbb{F}_q$ we can construct a quadratic nonresidue by finding elements of $\mathbb{F}_q$ corresponding to the sequence

$$(-1)^{\frac{1}{2}}, (-1)^{\frac{1}{4}}, \ldots, (-1)^{\frac{1}{2^{s-1}}}$$

for $s = \nu_2(q-1)$. In the other direction, given a quadratic nonresidue we can factor any degree two polynomial using Theorem 2.34. From Theorem 5.12 we have that given a finite field $\mathbb{F}_q$ and a field extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ so that an lth nonresidue is given in $\mathbb{F}_{q^k}$ we can unconditionally factor any polynomial of degree n so that l divides n (in $\text{poly}(\log(q), n^l)\mathbb{F}_p$-operations). If an algorithm $\mathcal{A}$ for completely factoring polynomials of degree $n \mid q-1$ is given, can $\mathcal{A}$ be used to construct an nth nonresidue in $\mathbb{F}_q$?

Question 5. In Chapter 4 and Chapter 5 we have made use of the composed quotient and product and the composed sum and difference respectively to study FPFF. Although more general frameworks for studying the factorization of polynomials over finite fields exist, the composed operations are useful because they are relatively concrete. Our use of these operations has been fairly straightforward. Can new results on FPFF be found by combining various composed operations such as the composed sum and the composed product?

References

[1] Nesmith C. Ankeny. The least quadratic non residue. *The Annals of Mathematics*, 55(1):65–72, 1952.

[2] Manuel Arora, Gábor Ivanyos, Marek Karpinski, and Nitin Saxena. Deterministic polynomial factoring and association schemes. page 68, 2012.

[3] Eric Bach, Joachim von zur Gathen, and Hendrik W Lenstra Jr. Factoring polynomials over special finite fields. *Finite Fields and Their Applications*, 7(1):5–28, 2001.

[4] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory: Volume 1 Efficient Algorithms*. The MIT Press, 1 edition, 1996.

[5] Elwyn R. Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, 1967.

[6] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[7] Joel V. Brawley, Shuhong Gao, and Donald Mills. Computing composed products of polynomials. *Contemporary Mathematics*, 225:1–15, 1999.

[8] David A Burgess. The distribution of quadratic residues and non-residues. *Mathematika*, 4(02):106–112, 1957.

[9] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[10] Qi Cheng and Ming-Deh A. Huang. Factoring polynominals over finite fields and stable colorings of tournaments. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 2000.

[11] Michele Cipolla. Sulla risoluzione apiristica delle congruenze binomie secondo un modulo primo. *Mathematische Annalen*, 63:54–61, 1907.

[12] Sergei Evdokimov. Factorization of polynomials over finite fields in subexponential time under GRH. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *Algorithmic number theory*, volume 877 of *Lecture Notes in Computer Science*, pages 209–219. Springer, 1994.

[13] Sergei Alekseevich Evdokimov. Factoring a solvable polynomial over a finite field and the Generalized Riemann Hypothesis. *Zapiski Nauchnykh Seminarov POMI*, 176:104–117, 1989.

[14] Philippe Flajolet, Alin Bostan, Bruno Salvy, and Éric Schost. Fast computation of special resultants. *J. Symb. Comput.*, 41(1):1–29, 2006.

[15] Shuhong Gao. On the deterministic complexity of factoring polynomials. *J. Symb. Comput.*, 31(1/2):19–36, 2001.

[16] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

[17] Gábor Ivanyos, Marek Karpinski, Lajos Rónyai, and Nitin Saxena. Trading GRH for algebra: Algorithms for factoring polynomials and related structures. *Mathematics of Computation*, 81(277):493–531, 2012.

[18] Gábor Ivanyos, Marek Karpinski, and Nitin Saxena. Schemes for deterministic polynomial factoring. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 191–198. ACM, 2009.

[19] Kiran S. Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011.

[20] Tanja Lange and Arne Winterhof. Factoring polynomials over arbitrary finite fields. *Theoretical Computer Science*, 234(1):301–308, 2000.

[21] Rudi Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, 2 edition, 2008.

[22] Ruediger Loos. Computing in algebraic extensions. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra*, pages 173–187. Springer, 1983.

[23] Brendan D. McKay. The asymptotic numbers of regular tournaments, Eulerian digraphs and Eulerian oriented graphs. *Combinatorica*, 10(4):367–377, 1990.

[24] Gary Mullen and Daniel Panario. *Handbook of Finite Fields*. Chapman and Hall/CRC, 2013.

[25] Jonathan Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation*, 55(192):745–763, 1990.

[26] Henry C Pocklington. The direct solution of the quadratic and cubic binomial congruences with prime moduli. In *Proc. Cambridge Phil. Soc*, volume 19, pages 57–59, 1917.

[27] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106110, 1978.

[28] Kenneth. Reid and Lowell Beineke. Tournaments. In Lowell Beineke and Robin Wilson, editors, *Selected Topics in Graph Theoy 1.* Academic Press, 1978.

[29] Lajos Rónyai. Factoring polynomials over finite fields. *Journal of Algorithms*, 9(3):391–400, 1988.

[30] Lajos Rónyai. Factoring polynomials modulo special primes. *Combinatorica*, 9(2):199–206, 1989.

[31] Chandan Saha. Factoring polynomials over finite fields using balance test. In Susanne Albers and Pascal Weil, editors, *STACS*, volume 1 of *LIPIcs*, pages 609–620. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[32] René Schoof. Elliptic curves over finite fields and the computation of square roots mod p?? *Mathematics of Computation*, 44(170):483–494, 1985.

[33] Victor Shoup. On the deterministic complexity of factoring polynomials over finite fields. *Information Processing Letters*, 33(5):261–267, 1990.

[34] Victor Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, pages 14–21. ACM, 1991.

[35] Igor Shparlinski. On finding primitive roots in finite fields. *Theor. Comput. Sci.*, 157(2):273–275, 1996.

[36] Tsz-Wo Sze. On taking square roots without quadratic nonresidues over finite fields. *Mathematics of Computation*, 80(275):1797–1811, 2011.

[37] Alberto Tonelli. Bemerkung uber die au osung quadratischer congruenzen. *Gottinger Nachrichten*, 32:344–46, 1891.

[38] Aleksandr Tuxanidy and Qiang Wang. Composed products and factors of cyclotomic polynomials over finite fields. *Des. Codes Cryptography*, 69(2):203–231.

[39] Joachim von zur Gathen. Factoring polynomials and primitive elements for special primes. *Theoretical Computer Science*, 52(1):77–89, 1987.

[40] Joachim von zur Gathen. Who was who in polynomial factorization: 1. In Barry M. Trager, editor, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2006, Genoa, Italy, July 9-12, 2006, Proceedings*, page 2. ACM, 2006.

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