REMOVING RANDOMNESS FROM COMPUTATIONAL NUMBER THEORY

by

VICTOR SHOUP

A thesis submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN — MADISON

1989

Abstract

In recent years, many probabilistic algorithms (i.e., algorithms that can toss coins) that run in polynomial time have been discovered for problems with no known deterministic polynomial time algorithms. Perhaps the most famous example is the problem of testing large (say, 100 digit) numbers for primality. Even for problems which are known to have deterministic polynomial time algorithms, these algorithms are often not as fast as some probabilistic algorithms for the same problem.

Even though probabilistic algorithms are useful in practice, we would like to know, for both theoretical and practical reasons, if randomization is really necessary to obtain the most efficient algorithms for certain problems. That is, we would like to know for which problems there is an inherent gap between the deterministic and probabilistic complexities of these problems.

In this research, we consider two problems of a number theoretic nature: factoring polynomials over finite fields and constructing irreducible polynomials of specified degree over finite fields. We present new results that narrow the gap between the known deterministic and probabilistic complexities of these problems. One of our results is a deterministic polynomial time reduction from the latter problem to the former, giving rise to a deterministic algorithm for constructing irreducible polynomials that runs in polynomial time for fields of small characteristic. Another of our results is a new deterministic factoring algorithm whose worst-case running time is asymptotically faster than that of previously known deterministic algorithms for this problem. We also analyze the average-case running time of our algorithm (averaging over inputs), proving that it is just about as fast as the expected running time (averaging over coin tosses) of some of the fastest probabilistic algorithms. In particular, the average-case running time of our algorithm is polynomial.

Preface

Some of the results in this thesis have previously appeared in [38] and [39].

I wish to thank my advisor, Eric Bach. Working with Eric has truly been a delight, and his encouragement and support were invaluable in conducting this research.

Thanks also goes to Anne Condon, Jon Sorenson and John Strikwerda for providing me with corrections and insightful comments on an early draft of this thesis.

I would like to also thank Hendrik Lenstra for sharing several of his ideas with me, and Hiroshi Gunji, in whose classes I learned quite a bit about algebra.

This research was supported by NSF grants DCR-8504485 and DCR-8552596.

Contents

Chapter 1

Introduction

The results in this thesis address the issue of the role of randomness in computation. There are many problems for which the best known deterministic algorithms are significantly slower than probabilistic algorithms for these problems; in some cases, this can mean the difference between an exponential time algorithm and a polynomial time algorithm. The question then arises: is randomness necessary in efficient algorithms for these problems? Instead of approaching this question from an abstract, complexity theoretic point of view, we approach it from a fairly concrete point of view: we study specific problems, examining the extent to which randomness can be avoided in efficient algorithms for their solution.

We study two computational problems of an algebraic nature and present results that narrow the gap between the known probabilistic and deterministic complexities of these problems. The problems studied are two fundamental problems in the theory of finite fields:

(1) factoring polynomials over a finite field, and

(2) constructing an irreducible polynomial of given degree over a finite field.

For the first problem, which is considered in Chapter 2, we present a new deterministic algorithm that is asymptotically faster than previously known deterministic algorithms for this problem. We also analyze the average-case complexity of our algorithm (assuming the polynomial to be factored is chosen at random from a uniform distribution) and prove that the average-case running time is polynomial. The average-case behavior of deterministic algorithms for this problem has not previously been studied.

For the second problem, considered in Chapter 3, we give the first deterministic algorithm for this problem that runs in polynomial time for finite fields of small characteristic. In fact, we prove the stronger result that the second problem can be reduced to the first problem deterministically in polynomial time, exhibiting a new connection between these two problems. We also show that for fields of small characteristic, this problem has a fast parallel algorithm (i.e., it is in the complexity class NC).

1.1 Probabilistic vs. Deterministic Algorithms

A *probabilistic algorithm* is simply an algorithm that is allowed to use random numbers. For such an algorithm, on a given input both the output and running time of the algorithm are random variables. For all probabilistic algorithms considered in this thesis, the output is guaranteed to be correct on all inputs. Furthermore, when we state the running time of a probabilistic algorithm, we are really stating the expected value of its running time.

Before proceeding any further, it is appropriate to address the question of why the deterministic complexity of factoring polynomials and finding irreducible polynomials should be studied at all, given that there are perfectly good probabilistic algorithms for their solution. Even though these probabilistic algorithms are quite adequate in practice, there are still compelling theoretical reasons to try to narrow the gap between the deterministic and probabilistic complexities of these problems.

First, it is an unresolved question as to whether the probabilistic model of computation is in fact more powerful than the traditional, deterministic model of computation. A line of research that examines this question as applied to specific problems seems bound to be more fruitful than a more general, abstract complexity-theoretic approach.

Second, the theory underlying probabilistic algorithms is adequate only to explain the behavior of a very idealized type of algorithm, and is not adequate to explain the behavior of probabilistic algorithms as they are implemented in practice. In theory, a probabilistic algorithm requires a source of independent random numbers; that is, the analysis of the algorithm assumes such a source. In practice, such a source is not available, and a deterministically computed sequence of "pseudorandom" numbers is used in actual implementations of the algorithm. In spite of this, these deterministic implementations exhibit behavior similar to that which would be expected had a source of independent random numbers been used.

Third, even assuming a source of independent random numbers is available, a probabilistic algorithm suffers from a lack of a *guaranteed* running time bound (we may be only able to bound its expected running time).

1.2 Background and Notation

A certain amount of knowledge of algebra and computation theory is presumed on the part of the reader. From algebra, the reader should be familiar with the definitions and basic properties of groups, rings, and fields (see, e.g., [41]). A knowledge of basic field theory—field extensions, norms and traces, Galois theory, finite fields—will be most helpful. For a comprehensive treatment of the theory of finite fields, see the book by Lidl and Niederreiter [30]. From the theory of computation, the reader should be familiar with the notions of computational complexity and polynomial time algorithms (see, e.g., [2]).

Although this thesis does not attempt to be self-contained, in the remainder of this section we collect in one place some basic facts from algebra and computation theory that are extensively used throughout this thesis, but that may not be familiar to the reader. But first, some notation. The expression $\log x$ denotes the logarithm to the base 2 of x, and $\ln x$ denotes the natural logarithm of x. Throughout this thesis, an expression of the form x^ϵ denotes a fixed but unspecified polynomial in $\log x$. This is somewhat nonstandard terminology, and if the reader prefers, he/she can interpret x^ϵ in the more traditional way as $x^{o(1)}$ without harm.

Algebraic Background

We shall now give a brief synopsis of some of the basic algebraic facts we will need. For proofs of the facts stated below, see [30].

Let R be a commutative ring with unity. Then R^* denotes the multiplicative group of units in R, and $R[X]$ denotes the ring of polynomials in one variable with coefficients in R.

Field Extensions

Let F be a field (finite or infinite). Then a polynomial in $F[X]$ is called *irreducible* if it cannot be written as the product of two polynomials of lower degree.

Suppose that F is a subfield of a larger field K. Then K is called a *field extension* of F. The *degree* of K over F, written $[K : F]$, is the dimension of K as a vector space over F. If E is a subfield of K that itself contains F as a subfield, then E is called an *intermediate* field between F and K, and we will sometimes call $K \supset E \supset F$ a *tower* of fields.

Lemma 1.1. *Let $K \supset E \supset F$ be a tower of fields. Then we have*

$$[K : F] = [K : E][E : F].$$

Now let K be a field extension of F, and let $\alpha \in K$. Then $F(\alpha)$ denotes the smallest subfield of K that contains all of the elements in F along with the element α. More generally, for $\alpha_1, \ldots, \alpha_m \in K$, $F(\alpha_1, \ldots, \alpha_m)$ denotes the smallest subfield of K that contains F and $\alpha_1, \ldots, \alpha_m$. The *degree* of α over F is defined to be $[F(\alpha) : F]$. In particular, $\alpha \in F$ if and only if the degree of α over F is 1. The element α is called *algebraic* over F if it is a zero of some polynomial over F.

Lemma 1.2. *Let K be a field extension of F with $\alpha \in K$ algebraic over F.*

(1) $F(\alpha) \cong F[X]/(f)$, where f is a uniquely determined monic irreducible polynomial in $F[X]$ with a zero at α.

(2) f is the monic polynomial of least degree with a zero at α; moreover, α is a zero of a polynomial $g \in F[X]$ if and only if g is divisible by f.

(3) If f has degree n, then $1, \alpha, \ldots, \alpha^{n-1}$ is a basis of $F(\alpha)$ over F. We have $[F(\alpha) : F] = n$ and each element of $F(\alpha)$ can be uniquely expressed as $a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1}$, where $a_i \in F$.

The polynomial f specified in this lemma is called the *minimum polynomial* of α over F. It plays a crucial role in our work.

In the situation above, we were given an element in an extension field of F, and then constructed an irreducible polynomial over F with a zero at α. We can turn this situation around. Suppose that we have a field F

and a monic irreducible polynomial f over F. Then we can construct the field $K = F[X]/(f)$. We can view K as an extension of F in a natural way, embedding F in K via the map $a \mapsto a \bmod f$, and in so doing we see that $K = F(\alpha)$ where α is the image of $X \bmod f$ in K. Also note that α is a root of f.

We will make some use of the language of Galois theory. Again, let K be a field extension over F. An automorphism σ on K (i.e., an isomorphism from K onto itself) is said to *fix* F if $a^\sigma = a$ for all $a \in F$ (we use superscript notation for application of automorphisms). It is easy to verify that the collection of automorphisms on K that fix F is a group with the group operation being function composition. This group is called the *Galois group* of K over F.

Finite Fields

A finite field F is simply a field with a finite number of elements. One can show that F contains $\mathbf{Z}_p$, the ring of integers modulo a prime p, as a subfield. In this situation, the prime p is uniquely determined by F and is called the *characteristic* of F.

Lemma 1.3. *Let F be a finite field of characteristic p.*

(1) F contains p^n elements, where $n = [F : \mathbf{Z}_p]$.

(2) The multiplicative group of units F^* is cyclic and contains $p^n - 1$ elements.

(3) Any other finite field with p^n elements is isomorphic to F.

One can also show that for every prime p and every positive integer n there is a finite field with p^n elements. By the previous lemma, this field is unique up to isomorphism, and we denote it by $\mathbf{F}_{p^n}$. The finite field $\mathbf{F}_{p^n}$ can be represented concretely as $\mathbf{F}_p[X]/(f)$ where f is any irreducible polynomial of degree n over $\mathbf{F}_p$. For fixed p, it is convenient to think of all of the fields $\mathbf{F}_{p^n}$ as lying in a single large field.

Now consider an arbitrary finite field $\mathbf{F}_q$, and let m be a positive integer. One can show that $\mathbf{F}_{q^m}$ contains $\mathbf{F}_q$ as a subfield. Let σ be the map on $\mathbf{F}_{q^m}$ given by $\alpha \mapsto \alpha^q$. σ is called the *Frobenius* map. The elements $\alpha, \alpha^\sigma, \ldots, \alpha^{\sigma^{m-1}}$ are called the *conjugates* of α (they may not be distinct).

Let T be the map on $\mathbf{F}_{q^m}$ given by $\alpha \mapsto \alpha + \alpha^\sigma + \cdots + \alpha^{\sigma^{m-1}}$. T is called the *trace* from $\mathbf{F}_{q^m}$ down to $\mathbf{F}_q$.

Lemma 1.4. *Assume the notation of the previous paragraph. Let $\alpha, \beta \in \mathbf{F}_{q^m}$.*

(1) $[\mathbf{F}_{q^m} : \mathbf{F}_q] = m$. *The intermediate fields between $\mathbf{F}_q$ and $\mathbf{F}_{q^m}$ are precisely $\mathbf{F}_{q^d}$ for d dividing m.*

(2) σ is an automorphism on $\mathbf{F}_{q^m}$ that fixes $\mathbf{F}_q$; moreover, the Galois group of $\mathbf{F}_{q^m}$ over $\mathbf{F}_q$ is cyclic of order m with generator σ.

(3) $\alpha \in \mathbf{F}_q$ if and only if $\alpha^\sigma = \alpha$; moreover, the degree d of α over $\mathbf{F}_q$ is the least d such that $\alpha^{\sigma^d} = \alpha$, and the minimum polynomial of α over $\mathbf{F}_q$ is $(X - \alpha)(X - \alpha^\sigma) \cdots (X - \alpha^{\sigma^{d-1}})$.

(4) T is an $\mathbf{F}_q$-linear map from $\mathbf{F}_{q^m}$ onto $\mathbf{F}_q$.

(5) $[\mathbf{F}_q(\alpha, \beta) : \mathbf{F}_q] = \mathrm{lcm}([\mathbf{F}_q(\alpha) : \mathbf{F}_q], [\mathbf{F}_q(\beta) : \mathbf{F}_q])$.

(6) The polynomial $X^{q^m} - X \in \mathbf{F}_q[X]$ is equal to the product of all of the monic irreducible polynomials over $\mathbf{F}_q$ whose degree divides m.

We will make use of the *Chinese Remainder Theorem* for polynomials, which we can formulate as follows.

Lemma 1.5. (Chinese Remainder Theorem) *Let $f_1, \ldots, f_r$ be distinct monic irreducible polynomials over $\mathbf{F}_q$, and let $g_1, \ldots, g_r$ be arbitrary polynomials over $\mathbf{F}_q$. Then the system of congruences $h \equiv g_i \pmod{f_i}$, $i = 1, \ldots, r$ has a unique solution h modulo $f_1 \cdots f_r$. Moreover,*

$$\mathbf{F}_q[X]/(f_1 \cdots f_r) \cong \mathbf{F}_q[X]/(f_1) \oplus \cdots \oplus \mathbf{F}_q[X]/(f_r),$$

where $g \bmod f_1 \cdots f_r \mapsto (g \bmod f_1, \ldots, g \bmod f_r)$ establishes the isomorphism.

Residues and Characters

Let $\mathbf{F}_q$ be a finite field. Let d divide $q - 1$, the order of the group $\mathbf{F}_q^*$. Then $\alpha \in \mathbf{F}_q^*$ is called a d-th *residue* if there exists $\beta \in \mathbf{F}_q^*$ such that $\beta^d = \alpha$, and is called a d-th *nonresidue* otherwise. Of particular interest is the case where q is odd and $d = 2$, in which case we speak of quadratic residues and nonresidues.

A multiplicative character on $\mathbf{F}_q$ is a map χ from $\mathbf{F}_q^*$ into the unit circle in the complex plane that satisfies $\chi(\alpha\beta) = \chi(\alpha)\chi(\beta)$ for all $\alpha, \beta \in \mathbf{F}_q^*$. It it easy to verify that $\chi(\alpha)$ is a $(q-1)$-th root of unity. The *trivial character* ε is defined by the relation $\varepsilon(\alpha) = 1$ for all $\alpha \in \mathbf{F}_q^*$. We will usually extend the domain of a character to all of $\mathbf{F}_q$. If $\chi \neq \varepsilon$, we do this by defining $\chi(0) = 0$. For ε, we define $\varepsilon(0) = 1$. The characters on $\mathbf{F}_q$ form a group with identity ε and multiplication of characters defined by $(\chi \cdot \psi)(\alpha) = \chi(\alpha)\psi(\alpha)$ for $\alpha \in \mathbf{F}_q^*$. In fact, the group of characters is cyclic of order $q-1$. The order d of a character χ is the least d such that $\chi^d = \varepsilon$. For odd q, we call the character of order 2 the quadratic character. The quadratic character takes on one of the values $0, \pm 1$.

Lemma 1.6. *Let $d \mid q-1$ and let χ be a character of order d.*

(1) $\alpha \in \mathbf{F}_q^*$ is a d-th nonresidue if and only if $\alpha^{(q-1)/d} \neq 1$ if and only if $\chi(\alpha) \neq 1$.

(2) For $\alpha \in \mathbf{F}_q$, the equation $X^d = \alpha$ has N solutions, where $N = \sum_{i=0}^{d-1} \chi^i(\alpha)$.

Computation Theory Background

The running time of algorithms discussed in this thesis are often expressed in terms of $\mathbf{F}_p$-operations, by which we mean one of the arithmetic operations $(+, -, *, \div)$ in the finite field $\mathbf{F}_p$ where p is a prime number. We gather together in one place the relevant facts about the asymptotic complexity of performing polynomial arithmetic.

Lemma 1.7. *Let R be a commutative ring with unity, and let F be a field. Let $L(n) = \log n \log\log n$.*

(1) Multiplication of two polynomials in $R[X]$ of degree $\leq n$ can be performed using $O(nL(n))$ operations ($+, -, \times$ only) in R.

(2) Let $\alpha_1, \ldots, \alpha_n \in R$. Then the coefficients of $(X - \alpha_1) \cdots (X - \alpha_n) \in R[X]$ can be computed using $O(nL(n)(\log n))$ operations ($+, -, \times$ only) in R.

(3) *Let f and g be polynomials in $R[X]$ of degree $\leq n$ and assume that the leading coefficient of g is a unit in R. Then $f \bmod g$ can be computed using $O(nL(n))$ operations in R.*

(4) *Let $f, g_1, \ldots, g_k$ be polynomials in $R[X]$ such that $\deg f \leq n$, $\deg g_1 + \cdots + \deg g_k \leq n$, and the leading coefficients of the g_i's are units in R. Then $f \bmod g_1, \ldots, f \bmod g_k$ can be computed using $O(nL(n)(\log n))$ operations in R.*

(5) *Let f be a polynomial in $R[X]$ of degree $\leq n$. Then f can be evaluated at the n points $\alpha_1, \ldots, \alpha_n \in R$ using $O(nL(n)(\log n))$ operations in R.*

(6) *Let f and g be polynomials in $F[X]$ of degree $\leq n$. Then the greatest common divisor d of f and g can be computed using $O(nL(n)(\log n))$ operations in F. Moreover, polynomials $a, b \in F[X]$ of degree $O(n)$ satisfying $af + bg = d$ can be computed in the same time bound.*

(7) *Let $\alpha \in R$. Then for any integer $m > 0$, α^m can be computed using $O(\log m)$ multiplications in R.*

(1) is proved in Cantor and Kaltofen [14]. We note that the results of Schönhage [37] would actually be sufficient for our purposes. (2) follows from (1) by a divide and conquer method (see [9, p. 100]). (3) follows from (1) by a Newton iteration scheme (see [9, p. 95]). (4) follows from (3) by a divide and conquer method (see [9, p. 100]). (5) follows from (4) by computing $f \bmod (X - \alpha_1), \ldots, f \bmod (X - \alpha_n)$. (6) follows from (1) by an algorithm described in [2, pp. 303–308]. (7) is proved using a simple repeated squaring algorithm.

Chapter 2

Factoring Polynomials over Finite Fields

2.1 Introduction

In this chapter, we consider the problem of factoring univariate polynomials over finite fields. This problem arises in many applications, including the construction of error correcting codes [7], the computation of discrete logarithms in finite fields [32], the factorization of multivariate polynomials over finite fields [46], and the factorization of polynomials over the integers [24, pp. 431–434].

Consider the problem of factoring a polynomial of degree n in $\mathbf{F}_p[X]$ where p is prime. There are several *deterministic* algorithms for this problem whose running time is polynomial for small p, more specifically, polynomial in n and p. One of the asymptotically fastest deterministic algorithms is based on a method of Berlekamp [8] as refined by von zur Gathen [45]. The Berlekamp–von zur Gathen algorithm requires $O(M(n) + pn^{2+\epsilon})$ $\mathbf{F}_p$-operations, where $M(n)$ is the number of $\mathbf{F}_p$-operations required to multiply two n by n matrices. Currently, the best known upper-bound on $M(n)$ is approximately $O(n^{2.4})$ [17].

There are also many *probabilistic* algorithms for this problem whose *expected* running time is polynomial, i.e. polynomial in n and $\log p$. One of the asymptotically fastest probabilistic algorithms is due to Ben-Or [6], which uses $O((\log p)n^{2+\epsilon})$ $\mathbf{F}_p$-operations. The running time of a different probabilistic algorithm due to Cantor and Zassenhaus [15] is also $O((\log p)n^{2+\epsilon})$ $\mathbf{F}_p$-operations. We should also mention that if $\log p$ is large with respect to n, another probabilistic algorithm of Cantor and Zassenhaus, which uses $O(M(n) + (\log p)n^{1+\epsilon})$ $\mathbf{F}_p$-operations, might be preferable to Ben-Or's.

This state of affairs suggests that there may be a significant gap between the deterministic and probabilistic complexity of this problem. Indeed, the running time of Ben-Or's probabilistic algorithm improves upon the running time of the Berlekamp–von zur Gathen deterministic algorithm with respect to both p and n. With respect to p, this improvement is exponential ($\log p$ vs. p), and it gives us an algorithm that runs in polynomial time for large p. With respect to n, this improvement is only polynomial ($n^{2+\epsilon}$ vs. $n^{2.4}$), but it could be substantial in cases where p is very small (e.g., $p = 2$) and n is very large.

We will give evidence that this gap is not quite so large. We present a new deterministic algorithm for factoring polynomials of degree n in $\mathbf{F}_p[X]$, and prove the following results about its running time (expressed in terms of $\mathbf{F}_p$-operations):

(1) *The worst-case running time is* $O((\log p)n^{2+\epsilon} + p^{1/2}(\log p)^2 n^{3/2+\epsilon})$.

(2) *The fraction of polynomials of degree n over $\mathbf{F}_p$ for which our algorithm fails to halt in time* $O((\log p)n^{2+\epsilon} + (\log p)^2 n^{1+\epsilon})$ *is* $O((\log n \cdot \log p)^2/p)$.

(3) *The average-case running time (assuming the input is drawn from a uniform distribution on all monic polynomials of degree n over $\mathbf{F}_p$) is* $O((\log p)n^{2+\epsilon})$.

Result (1) is significant for two reasons. First, if p is very small (which is in fact the case in applications such as constructing error correcting codes and factoring polynomials over the integers), the dependence on n in the running time becomes the dominant factor. The Berlekamp–von zur Gathen algorithm requires the triangularization of an n by n matrix, leading to the $M(n)$ term in the running time. Ben-Or's algorithm avoids the need to do this linear algebra by using randomization. Our algorithm also avoids the need to do any linear algebra, but without resorting to randomization.

Second, if p is very large, the dependence on p in the running time becomes the dominant factor. The Berlekamp–von zur Gathen algorithm requires a deterministic search through potentially all of $\mathbf{F}_p$. Ben-Or's algorithm replaces this brute-force search with a fast random search. Our algorithm performs a deterministic search, but we prove that the length of this search is bounded by $p^{1/2}(\log p)$. The dependence on p in our algorithm, though exponentially worse than that of Ben-Or's, is still significantly better than that of the Berlekamp–von zur Gathen algorithm.

Result (2) shows that our algorithm runs in polynomial time for all but at most an exponentially small fraction of polynomials of degree n over $\mathbf{F}_p$. One could conjecture that our algorithm in fact runs in polynomial time for all inputs, but proving this appears to be very hard; our result at least gives some quantitative evidence in support of such a conjecture.

Result (3) appears to be the first analysis of the average-case complexity of an algorithm for factoring polynomials over finite fields. Not only does it show that the problem of factoring polynomials has a polynomial time average-case complexity, but it also shows that the expected running time of Ben-Or's probabilistic algorithm (averaging over coin tosses) and the expected running time of our deterministic algorithm (averaging over inputs) are roughly the same (to within a factor of $(\log n)^{O(1)}$).

Our algorithm is fairly simple, and the space requirement of our algorithm is approximately that needed for $O(n^{2+\epsilon})$ field elements. The analysis of the dependence on n in the running time of our algorithm relies on a new application of fast algorithms for multiplying polynomials over a ring. The worst-case and average-case analysis of the dependence on p in the running time of our algorithm makes use of estimates of the number of solutions to equations over finite fields; similar techniques have been previously used in the analysis of various probabilistic algorithms [4, 5, 6].

We also mention another deterministic factoring algorithm (conveyed to the author by Lenstra [29]) that uses a "baby-step giant-step" method to obtain a $p^{1/2} \cdot (n \log p)^{O(1)}$ running time bound; however, the algorithm requires time and space $p^{1/2} \cdot (n \log p)^{O(1)}$ even in the best case. Since this algorithm is apparently not accessible in the literature, we briefly describe it in the next section.

The rest of this chapter is organized as follows. In Section 2.2, we give a brief overview of some well-known (and some not so well-known) factoring methods. In Section 2.3, we describe our new factoring algorithm, and analyze its worst-case running time. In Section 2.4, we analyze the average-case complexity of our new factoring algorithm. Finally, in Section 2.5, we close with some open questions.

2.2 A Brief Overview of Factoring Methods

In this section, we briefly review some of the key ideas found in algorithms for factoring polynomials over $\mathbf{F}_p$. For more complete surveys of this area, see [30, Chapter 4] and also [28]. The notation and terminology introduced in this section will be used in the remaining sections of this chapter.

Let $f \in \mathbf{F}_p[X]$ be a monic polynomial of degree n that we wish to factor. As a first step in factoring f, many algorithms perform what is called "distinct degree factorization" [13, 15]. That is, we construct polynomials $f^{(1)}, \ldots, f^{(n)}$ where $f^{(d)}$ $(1 \leq d \leq n)$ is the product of all the distinct monic irreducible polynomials of degree d that divide f. Using algorithms described in [13] and [15], distinct degree factorization can be performed using $O((\log p)n^{2+\epsilon})$ $\mathbf{F}_p$-operations. These algorithms make use of the fact that $X^{p^d} - X$ is the product of all monic irreducible polynomials whose degree divides d. One computes $h = X^p - X \bmod f$, and then $f^{(1)} = \gcd(f, h)$. After removing all linear factors from f (by repeatedly dividing f by $\gcd(f, h)$ as necessary), we compute $h = X^{p^2} - X \bmod f$, and then $f^{(2)} = \gcd(f, h)$. We can proceed in this fashion to obtain $f^{(3)}, \ldots, f^{(n)}$. For any d, $1 \leq d \leq n$, this procedure not only gives us $f^{(d)}$, but it also gives us the number of irreducible factors of $f^{(d)}$ (which is just $\deg f^{(d)}/d$).

It is possible that each $f^{(d)}$ is itself irreducible, in which case we are done; however, in the general case, some $f^{(d)}$ will require further factorization.

Let $1 \leq d \leq n$ be fixed and let $g = f^{(d)}$. We want to factor g. Suppose $g = g_1 \cdots g_k$, where the g_i's are distinct monic irreducible polynomials of degree d. We can assume that $k > 1$. Let $m = \deg g = kd$. Also, let $R = \mathbf{F}_p[X]/(g)$ and $x = X \bmod g \in R$. Finally, let θ_i be the natural homomorphism from the $\mathbf{F}_p$-algebra R onto the extension field $\mathbf{F}_p[X]/(g_i)$ of $\mathbf{F}_p$. By the Chinese Remainder Theorem, the map which takes $\alpha \in R$ to $(\theta_1(\alpha), \ldots, \theta_k(\alpha))$ is an isomorphism of the $\mathbf{F}_p$-algebras R and $\mathbf{F}_p[X]/(g_1) \oplus \cdots \oplus \mathbf{F}_p[X]/(g_k)$.

Following Camion [13], the *Berlekamp subalgebra* B of R is defined by $B = \{\alpha \in R : \theta_i(\alpha) \in \mathbf{F}_p \text{ for each } i = 1, \ldots, k\}$. Note that the Chinese Remainder Theorem gives an isomorphism between B and $\mathbf{F}_p \oplus \cdots \oplus \mathbf{F}_p$. Also following Camion, we call a subset $S \subset B$ a *separating set* if for any $1 \leq i < j \leq k$ there exists $s \in S$ such that $\theta_i(s) \neq \theta_j(s)$.

All of the efficient deterministic algorithms for factoring over $\mathbf{F}_p$ involve the computation of a small separating set. The reason we want a separating set is the following. To obtain a complete factorization of g, for each pair $1 \le i < j \le k$, it is sufficient to find a factor of g that "separates" g_i and g_j—i.e., is divisible by exactly one of g_i or g_j. To do this, it suffices to find $\alpha \in R$ such that exactly one of $\theta_i(\alpha)$ and $\theta_j(\alpha)$ is zero. For such an α, $\gcd(g, \alpha)$ will separate g_i and g_j, since $\alpha \equiv 0 \pmod{g_i}$ or $\alpha \equiv 0 \pmod{g_j}$, but not both. Given a separating set S, we know that for some $s \in S$, $\theta_i(s) = a$ and $\theta_j(s) = b$, where a and b are distinct elements of $\mathbf{F}_p$. For some $\delta \in \mathbf{F}_p$, which we can find by trying $\delta = 0, 1, 2$, etc., we will have $a + \delta = 0$ or $b + \delta = 0$, but not both. For this choice of δ, $\gcd(g, s + \delta)$ separates g_i and g_j, since $\theta_i(s + \delta) = 0$ or $\theta_j(s + \delta) = 0$, but not both. If both p and the size of S are small, then this approach will lead to an efficient deterministic factoring algorithm.

Constructing a Separating Set

It should be emphasized that the Chinese Remainder isomorphism is used only in the analysis of factoring algorithms—not in their implementation. Indeed, computing this isomorphism is equivalent to the original factoring problem (up to polynomial time reductions). In particular, the problem of finding a separating set is not just a simple combinatorics problem. In fact, when p is small, the computation of a separating set is the most expensive operation in all known deterministic factoring algorithms.

Berlekamp's method for computing a separating set is based on the observation that B is the kernel of the $\mathbf{F}_p$-linear map on R that takes α to $\alpha^p - \alpha$. Using techniques from linear algebra, we can compute a basis for B, which forms a separating set of k elements. The complexity of computing this basis is $O((\log p)m^{1+\epsilon} + m^3)$ $\mathbf{F}_p$-operations if classical algorithms of linear algebra are used, and about $O((\log p)m^{1+\epsilon} + m^{2.4})$ $\mathbf{F}_p$-operations if asymptotically fast algorithms for matrix arithmetic are used.

Another approach for computing a separating set is described by Camion [13]. Let T be the $\mathbf{F}_p$-linear map on R that takes α to $\alpha + \alpha^p + \cdots + \alpha^{p^{d-1}}$. Note that for any i, $\theta_i(T(\alpha))$ is just the trace from $\mathbf{F}_p[X]/(g_i)$ down to $\mathbf{F}_p$ of $\theta_i(\alpha)$. By viewing T as an $\mathbf{F}_p$-linear map on $\mathbf{F}_p[X]/(g_i g_j)$, and noting that the residues mod $g_i g_j$ of $1, X, \ldots, X^{2d-1}$ form a basis for this algebra over $\mathbf{F}_p$, one can easily show that $S = \{T(x), T(x^2), \ldots, T(x^{2d-1})\}$ is a separating set.

Making use of the algorithm described in [13], one can compute S with

$O((\log p)dm^{1+\epsilon} + d^2m^{1+\epsilon})$ $\mathbf{F}_p$-operations. Therefore, the worst-case complexity of Camion's algorithm is $O((\log p)m^{2+\epsilon} + m^{3+\epsilon})$. Actually, Camion's algorithm successively computes $T(x), T(x^2)$, etc., until a separating set is obtained. So in some cases we might not have to compute the entire set S—we might be able to get by with a smaller set $S_e = \{T(x), T(x^2), \ldots, T(x^e)\}$, where $e \leq 2d-1$, which can be computed at a cost of only $O((\log p)dm^{1+\epsilon} + edm^{1+\epsilon})$ $\mathbf{F}_p$-operations.

Camion did not prove any lower bounds on how big e needs to be in order for S_e to be a separating set. The following argument shows that in some cases e may have to be $\Omega(d)$. Again, view T as an $\mathbf{F}_p$-linear map on $\mathbf{F}_p[X]/(g_ig_j)$; further, suppose that $g_i = X^d + a_1X^{d-1} + \cdots + a_d$, $g_j = X^d + b_1X^{d-1} + \cdots + b_d$, and μ is the least integer such that $a_\mu \neq b_\mu$. Then it follows from Lemma 2.1 below that $e \geq \mu$ is a necessary condition for S_e to be a separating set for g; and furthermore, if $\mu < p$, then this is a sufficient condition for S_e to be a separating set for g_ig_j. So if g is divisible by a pair of irreducible polynomials g_i, g_j, whose high order coefficients agree to $\Omega(d)$ places, then Camion's algorithm will require $\Omega((\log p)dm^{1+\epsilon} + d^2m^{1+\epsilon})$ $\mathbf{F}_p$-operations. Such pairs of irreducible polynomials certainly exist; for example, results in the next chapter imply that such pairs of polynomials exist for all odd primes p and all d which are powers of 2 (see Section 3.2, Step 1, Case 2). Therefore, the worst-case complexity of Camion's algorithm for constructing a separating set is $\Omega((\log p)m^{2+\epsilon} + m^{3+\epsilon})$, which is certainly worse than the complexity of Berlekamp's algorithm.

The argument in the previous paragraph made use of the following lemma:

Lemma 2.1. *If γ is an algebraic element over $\mathbf{F}_p$ with minimum polynomial $X^d + c_1X^{t-1} + \cdots + c_d$, and T is the trace from $\mathbf{F}_p(\gamma)$ down to $\mathbf{F}_p$, then*

$$\begin{aligned}
T(\gamma) + c_1 &= 0\\
T(\gamma^2) + c_1T(\gamma) + 2c_2 &= 0\\
T(\gamma^3) + c_1T(\gamma^2) + c_2T(\gamma) + 3c_3 &= 0\\
&\vdots\\
T(\gamma^d) + c_1T(\gamma^{d-1}) + \cdots c_{d-1}T(\gamma) + dc_d &= 0
\end{aligned}$$

Proof. This follows immediately from Newton's formulas for sums of powers of the roots of a polynomial [40, p. 261]. $\square$

In the next section, we will present a deterministic algorithm that constructs a separating set of d elements that uses only $O((\log p)dm^{1+\epsilon})$ $\mathbf{F}_p$-operations. This is how we reduce the dependence on n in the deterministic complexity of factoring polynomials over $\mathbf{F}_p$ from $n^{2.4}$ to $n^{2+\epsilon}$.

Using a Separating Set

Once we have a separating set for g, we could use it as described above to factor g. That is, to separate g_i and g_j, we can choose s in S with $\theta_i(s) = a$ and $\theta_j(s) = b$, where a and b are distinct elements in $\mathbf{F}_p$, and then search for $\delta \in \mathbf{F}_p$ for which $a + \delta = 0$ or $b + \delta = 0$. If we just search for an appropriate value of δ by examining $\delta = 0, 1, 2$, etc., this can take time proportional to p, which is quite impractical if p is very large.

One way to approach this problem, based on an idea originating in [8], is to compute $\beta = (s + \delta)^{(p-1)/2}$ for various choices of δ. $\theta_i(\beta)$ is just $\chi(a + \delta)$, where χ is the quadratic character on $\mathbf{F}_p$; likewise, $\theta_j(\beta)$ is $\chi(b+\delta)$. Thus, if $\chi(a+\delta) \neq \chi(b+\delta)$, then either $\gcd(g, \beta-1)$ or $\gcd(g, \beta)$ separates g_i and g_j (since the values of $\theta_i(\beta)$ and $\theta_j(\beta)$ are either 0 or ± 1). So we must search for $\delta \in \mathbf{F}_p$ such that $\chi(a + \delta) \neq \chi(b + \delta)$.

One search strategy is to randomize: simply choose successive values of $\delta \in \mathbf{F}_p$ at random. One can easily show that for fixed $a \neq b$ and randomly chosen δ, the probability that $\chi(a + \delta) = \chi(b + \delta)$ is no more than $1/2$. So with high probability, g_i and g_j can be quickly separated. This idea is more fully developed in [6] and [27].

Another search strategy is a simple brute-force deterministic search: examine $\delta = 0, 1, 2$, etc., until we find a value of δ for which $\chi(a+\delta) \neq \chi(b+\delta)$. In [8], Berlekamp suggested a strategy similar to this, but did not analyze its complexity. We will prove that for some δ, with $0 \leq \delta \leq p^{1/2}\log p$, we have $\chi(a + \delta) \neq \chi(b + \delta)$. This bound on δ is very crude—there are intuitive reasons to believe that $\chi(a + \delta) \neq \chi(b + \delta)$ for much smaller values of δ. But as a lower bound on δ, Camion [12] shows (using an elementary counting argument) that for all p there exist $a, b \in \mathbf{F}_p$, where $a \neq b$, such that $\chi(a + \delta) = \chi(b + \delta)$ for all $0 \leq \delta \leq \log p - 2$. However, we will show that such "bad pairs" a, b are actually quite rare, proving that the number of pairs $a, b \in \mathbf{F}_p$ such that $\chi(a + \delta) = \chi(b + \delta)$ for all $0 \leq \delta \leq \lceil \log p \rceil - 1$ is $O(p \log p)$. This fact will then be used to show that the average-case complexity of our algorithm is polynomial, assuming that the coefficients of the input polynomial are chosen at random.

We should also remark that in [8], Berlekamp mentions the result of Burgess [10] that the maximum number of consecutive quadratic residues or nonresidues mod p is $O(p^{1/4+\epsilon})$; unfortunately, this result by itself has no bearing on the complexity of factoring polynomials. The relevant question is: what is the maximum number of consecutive quadratic residues in the sequence $\{(a+\delta)(b+\delta)\}_{\delta\geq 0}$? For *fixed* $a \neq b$, later results of Burgess [11] imply that the maximum number of consecutive quadratic residues or nonresidues in this sequence is $O(p^{1/4+\epsilon})$; however, the "constant" implied by the big-"O" depends on a and b, and an examination of Burgess' proof reveals that this dependence is proportional to $|a-b|$. Thus, this result also has no immediate bearing on the complexity of factoring polynomials.

Another $p^{1/2}$ Factoring Algorithm

Lenstra [29] describes a completely different method to factor polynomials in time $p^{1/2} \cdot (n \log p)^{O(1)}$. This algorithm requires time and space $p^{1/2} \cdot (n \log p)^{O(1)}$ even in the best case. We shall briefly describe this algorithm here. By results in [8], the problem of factoring a polynomial deterministically reduces in time $(n \log p)^{O(1)}$ to the problem of finding the zeros of a polynomial. Suppose we want to find the zeros of a polynomial $f \in \mathbf{F}_p[X]$ of degree n. Let $t = \lfloor \sqrt{p} \rfloor$. Then it will suffice to calculate the gcd of f with each of the following polynomials:

$$z_i(X) = (X - it)(X - (it+1)) \cdots (X - (it + (t-1))) \qquad (i = 0, \ldots, t-1).$$

If a nontrivial gcd is found, we can search in an interval of length t for zeros of f. The remaining elements $t^2, \ldots, p-1$ can be checked separately.

To calculate the z_i, note that we only need to calculate them mod f, i.e. in the ring $R = \mathbf{F}_p[X]/(f)$. To do this, we compute the coefficients of the polynomial $h(Y) \in \mathbf{F}_p[Y]$, where $h(Y) = Y(Y-1)\cdots(Y-(t-1))$. By Lemma 1.7, this can be done in time $p^{1/2} \cdot (n \log p)^{O(1)}$. Let $x = X \bmod f$ be the image of X in R. Then $z_i(x) = h(x - it)$ for $i = 0, \ldots, t-1$. So to compute all of the $z_i(x)$'s, it suffices to evaluate the polynomial $h(Y)$ at t points. But this can be done in time $p^{1/2} \cdot (n \log p)^{O(1)}$ (again by Lemma 1.7). So the entire algorithm takes time $p^{1/2} \cdot (n \log p)^{O(1)}$.

2.3 A New Factoring Algorithm

In this section, we describe a new algorithm for factoring polynomials over $\mathbf{F}_p$, and analyze its worst-case running time. As in the previous section, let $f \in \mathbf{F}_p[X]$ be a polynomial of degree n that we wish to factor. We first perform distinct degree factorization. Now let $1 \leq d \leq n$ be fixed, and let $g = f^{(d)}$. We want to factor $g = g_1 \cdots g_k$, where the g_i's are distinct monic irreducible polynomials of degree d. As in the last section, $m = \deg g = kd$, $R = \mathbf{F}_p[X]/(g)$, $x = X \bmod g \in R$, and θ_i is the natural homomorphism from R onto $\mathbf{F}_p[X]/(g_i)$.

To factor g, our algorithm first computes a separating set in the following way. We compute the coefficients of $h(Y) \in R[Y]$ where $h(Y) = (Y - x)(Y - x^p) \cdots (Y - x^{p^{d-1}})$. Suppose $h(Y) = Y^d + s_1 Y^{d-1} + \cdots + s_d$, where $s_i \in R$. We claim that $\{s_1, \ldots, s_d\}$ is a separating set. To see this, first observe that θ_i extends in a natural way to a homomorphism from $R[Y]$ onto $(\mathbf{F}_p[X]/(g_i))[Y]$ by coefficientwise application of θ_i. Next observe that for each $i = 1, \ldots, k$,

$$
\begin{aligned}
\theta_i(h(Y)) &= Y^d + \theta_i(s_1)Y^{d-1} + \cdots + \theta_i(s_d) \\
&= (Y - \theta_i(x))(Y - \theta_i(x)^p) \cdots (Y - \theta_i(x)^{p^{d-1}}) \\
&= g_i(Y),
\end{aligned}
$$

the last equality following from the fact that $\theta_i(x), \ldots, (\theta_i(x))^{p^{d-1}}$ are the roots of $g_i(Y)$ in the field $\mathbf{F}_p[X]/(g_i)$. Our claim now follows immediately from the fact that the g_i are pairwise distinct.

Now that we have a separating set, we can proceed to factor g. We construct finer and finer partial factorizations $U \subset \mathbf{F}_p[X]$ consisting of monic polynomials with $\prod_{u \in U} u = g$. Initially, we put $U = \{g\}$. We make use of the operation $Refine(U, v)$, which, when given a partial factorization U and a polynomial $v \in \mathbf{F}_p[X]$, produces the refinement of U given by $\bigcup_{u \in U} \{\gcd(u, v), u/\gcd(u, v)\} \backslash \{1\}$. We will also make use of the function $Test(U, v)$, which, when given a partial factorization U and a polynomial $v \in \mathbf{F}_p[X]$, returns *true* if $v \bmod u \notin \mathbf{F}_p$ for some $u \in U$, and *false* otherwise. Intuitively, $Test(U, v)$ determines if v is of any value in obtaining further refinements of U. To obtain a complete factorization of g, we execute the refinement procedure in Figure 2.1.

The correctness of this algorithm follows from the discussion in the previous section, along with the fact the $\{s_1, \ldots, s_d\}$ is a separating set. The

```
i ← 1
while |U| < k do
    δ ← 0
    while Test(U, s_i) do
        U ← Refine(U, s_i + δ)
        if p ≠ 2 then U ← Refine(U, (s_i + δ)^((p−1)/2) − 1)
        δ ← δ + 1
    i ← i + 1
```

Figure 2.1: Refinement Procedure

order in which the s_i are utilized in the refinement procedure is immaterial as far as the worst-case analysis is concerned; however, the assumption that s_1 is utilized first is crucial in the average-case analysis. To analyze the worst-case running time of our algorithm, we shall make use of the following two lemmas.

Lemma 2.2. *Let $\mathbf{F}_q$ be the finite field with q elements, let ψ be a multiplicative character of order t on $\mathbf{F}_q$, and let λ be a monic polynomial in $\mathbf{F}_q[X]$ that is not a perfect t-th power. Suppose further that λ has r distinct roots in its splitting field. Then*

$$\left|\sum_{x\in\mathbf{F}_q} \psi(\lambda(x))\right| \le (r-1)q^{1/2}.$$

For a proof of this lemma, see [30, p. 225] or [36, p. 43].

Lemma 2.3. *Let p be an odd prime, and let $a, b \in \mathbf{F}_p$, such that $a \neq b$ and let χ be the quadratic character on $\mathbf{F}_p$. Then there exists δ, with $0 \le \delta \le p^{1/2}\log p$, such that $\chi(a+\delta) \neq \chi(b+\delta)$.*

Proof. Let $t = \lceil(\log p)/2\rceil$. Let N be the number of solutions $(x, y_0, \ldots, y_{t-1}) \in \mathbf{F}_p^{t+1}$ to the system of equations

$$(x+a+i)(x+b+i) = y_i^2 \qquad (i = 0, \ldots, t-1).$$

We will first show that

$$N \le p + p^{1/2}(2^t(t-1)+1). \tag{$*$}$$

Now, for fixed $c \in \mathbf{F}_p$ the number of $y \in \mathbf{F}_p$ satisfying the equation $y^2 = c$ is precisely $1 + \chi(c)$. Therefore,

$$\begin{aligned} N &= \sum_{x \in \mathbf{F}_p} \prod_{i=0}^{t-1} (1 + \chi((x+a+i)(x+b+i))) \\ &= \sum_{0 \le e_0, \ldots, e_{t-1} \le 1} \quad \sum_{x \in \mathbf{F}_p} \chi\Big(\prod_{i=0}^{t-1} (x+a+i)^{e_i} (x+b+i)^{e_i} \Big). \end{aligned}$$

In this last expression, the term corresponding to $e_0 = \cdots = e_{t-1} = 0$ is p.

Now let $e_0, \ldots, e_{t-1}$ be fixed with $l > 0$ of the e_i's are nonzero, and let $\lambda(X) = \prod_{i=0}^{t-1}(X + a + i)^{e_i}(X + b + i)^{e_i}$. We claim that $\lambda(X)$ is not a perfect square in $\mathbf{F}_p[X]$. Suppose that it were. Then for distinct $i_1, \ldots, i_l$ between 0 and $t - 1$, we would have

$$a + i_1 = b + i_2, \quad a + i_2 = b + i_3, \quad \ldots, \quad a + i_{l-1} = b + i_l, \quad a + i_l = b + i_1.$$

Summing, we have $la + \sum_\nu i_\nu = lb + \sum_\nu i_\nu$. But this implies that $la = lb$, and since $0 < l < p$, we can cancel, obtaining $a = b$, a contradiction. Therefore, $\lambda(X)$ is not a perfect square, and so we can use the previous lemma to obtain

$$\begin{aligned} N &\le p + p^{1/2} \sum_{l=1}^{t} \binom{t}{l} (2l - 1) \\ &= p + p^{1/2}(2^t(t-1) + 1). \end{aligned}$$

This proves $(*)$.

Now, the number of $x \in \mathbf{F}_p$ such that

$$\chi((x+a+i)(x+b+i)) = 1 \qquad (i = 0, \ldots, t-1) \qquad (**)$$

is at most $N/2^t$.

Let δ be the least nonnegative integer such that $\chi(a + \delta) \ne \chi(b + \delta)$. The worst possible case is when all x satisfying $(**)$ are bunched together near zero. It follows that $\delta \le N/2^t + t$. By $(*)$, we have $\delta \le p/2^t + p^{1/2}(t - 1 + 2^{-t}) + t$. Since $t = \lceil (\log p)/2 \rceil$, we have $\delta \le p^{1/2} + (p^{1/2} \log p)/2 + (\log p)/2 + 2$. The right hand side of this inequality is asymptotic to $(p^{1/2} \log p)/2$, and is less than $p^{1/2} \log p$ for $p > 16$. For $p < 16$, $p^{1/2} \log p > p$, in which case the lemma is trivially true. □

The following theorem bounds the worst-case running time of our algorithm.

Theorem 2.4. *Let f be a polynomial of degree n in $\mathbf{F}_p[X]$. Then our algorithm will completely factor f using $O((\log p)n^{2+\epsilon} + p^{1/2}(\log p)^2 n^{3/2+\epsilon})$ $\mathbf{F}_p$-operations.*

Proof. The distinct degree factorization of f can be performed using $O((\log p)n^{2+\epsilon})$ $\mathbf{F}_p$-operations.

Consider factoring $g = f^{(d)}$ for a fixed $1 \leq d \leq n$. Our algorithm can construct the separating set $S = \{s_1, \ldots, s_d\}$ using $O((\log p)d)$ multiplications in R to compute the powers of x, and $O(d^{1+\epsilon})$ additions, multiplications and subtractions in R to compute the coefficients of $h(Y)$. Since each R-operation takes $O(m^{1+\epsilon})$ $\mathbf{F}_p$-operations, this gives a total of $O((\log p)dm^{1+\epsilon})$ $\mathbf{F}_p$-operations to compute S.

Now, for any partial factorization U and any polynomial v of degree less than m, we can compute $Refine(U, v)$ with $O(m^{1+\epsilon})$ $\mathbf{F}_p$-operations by first computing $v \bmod u$ for each $u \in U$, and then computing $\gcd(u, v \bmod u)$ for each $u \in U$. Similarly, we can compute $Test(U, v)$ with $O(m^{1+\epsilon})$ $\mathbf{F}_p$-operations. The computation of $(s_i + \delta)^{(p-1)/2}$ takes $O((\log p)m^{1+\epsilon})$ $\mathbf{F}_p$-operations using a fast exponentiation algorithm. Therefore, each execution of the body of the inner loop of the refinement procedure (lines 4–7 in Figure 2.1) takes $O((\log p)m^{1+\epsilon})$ $\mathbf{F}_p$-operations, and each termination test of this loop takes $O(m^{1+\epsilon})$ $\mathbf{F}_p$-operations.

Now consider the i-th iteration of the outer loop. For this iteration of the outer loop, the body of the inner loop is executed some number, say M_i, times. First observe that the number of i $(1 \leq i \leq d)$ for which $M_i > 0$ is at most k, since whenever $M_i > 0$, the i-th iteration of the outer loop will produce a finer factorization, and this can happen at most k times. By the previous lemma, if M_i is nonzero, it is at most $p^{1/2}(\log p) + 1$ when p is odd. Obviously, this bound holds for $p = 2$ as well, so we can completely factor g using $O((\log p)dm^{1+\epsilon} + p^{1/2}(\log p)^2 \min(d, k)m^{1+\epsilon})$ $\mathbf{F}_p$-operations.

Since $\min(d, k) \leq m^{1/2}$, the number of $\mathbf{F}_p$-operations required to factor $g = f^{(d)}$ is $O((\log p)dm^{1+\epsilon} + p^{1/2}(\log p)^2 m^{3/2+\epsilon})$. Since this holds for all $1 \leq d \leq n$, the number of $\mathbf{F}_p$-operations required to factor f is $O((\log p)n^{2+\epsilon} + p^{1/2}(\log p)^2 n^{3/2+\epsilon})$. $\square$

Example. We will illustrate the workings of our algorithm with an example. Let $p = 1061$ and consider the polynomial

$$f = X^6 + 15X^5 + 48X^4 + 295X^3 + 300X^2 + 267X + 19.$$

We will use our new algorithm to factor f. As it happens, f factors as $f = f_1 f_2 f_3$, where

$$f_1 = X^2 + X + 1, \quad f_2 = X^2 + X + 19, \quad f_3 = X^2 + 13X + 1.$$

The algorithm first performs the distinct degree factorization procedure. This procedure first computes $\gcd(X^p - X, f)$. Finding this to be 1 (since f has no linear factors), this procedure then computes $\gcd(X^{p^2} - X, f)$. Finding this to be f, the procedure terminates, concluding that f is the product of 3 distinct monic irreducible polynomials each of degree 2.

Let $R = \mathbf{F}_p[X]/(f)$, and let $x = X \bmod f \in R$. By the Chinese Remainder Theorem, we know that R is isomorphic to $\mathbf{F}_{p^2} \oplus \mathbf{F}_{p^2} \oplus \mathbf{F}_{p^2}$. Under this isomorphism, x corresponds to (α, β, γ), where $\alpha, \beta, \gamma \in \mathbf{F}_{p^2}$ are the roots of the irreducible polynomials f_1, f_2, f_3, respectively.

The algorithm now computes a separating set. It does this by computing the coefficients of $h(Y) = (Y - x)(Y - x^p) \in R[Y]$. Upon performing this computation, we find that $h(Y) = Y^2 + s_1 Y + s_2$, where

$$s_1 = 173x^5 + 644x^4 + 1046x^3 + 169x^2 + 759x + 358,$$

and

$$s_2 = 890x^5 + 892x^4 + 999x^3 + 212x^2 + 210x + 104.$$

Under the isomorphism given by the Chinese Remainder Theorem, $h(Y)$ corresponds to

$$(Y - (\alpha, \beta, \gamma))(Y - (\alpha^p, \beta^p, \gamma^p)) = Y^2 + \underbrace{(1, 1, 13)}_{s_1} Y + \underbrace{(1, 19, 1)}_{s_2}.$$

The algorithm now proceeds to factor f given the separating set $\{s_1, s_2\}$. It initializes U to $\{f\}$. The algorithm computes $\gcd(f, s_1)$. Finding this to be 1, the algorithm then computes $f' = \gcd(f, s_1^{(p-1)/2} - 1)$. It turns out that $f' = f_1 f_2$, and so U becomes $\{f', f_3\}$. But now the algorithm realizes that no further use can be made of s_1, since $s_1 \bmod f' = 1 \in \mathbf{F}_p$, and moves on to s_2. To see why $f' = f_1 f_2$, note that that 13 is a quadratic nonresidue mod 1061, and so $s_1^{(p-1)/2} - 1$ corresponds to $(0, 0, -2)$. Therefore, $\gcd(f, s_1^{(p-1)/2} - 1) = f_1 f_2$.

The algorithm now computes $\gcd(f', s_2)$. Finding this to be 1, the algorithm then computes $\gcd(f', s_2^{(p-1)/2} - 1)$. It turns out that the value of this gcd is just f'. To see why this is the case, note that $19 \equiv 329^2 \pmod{1061}$, and so $s_2^{(p-1)/2}-1$ corresponds to $(0,0,0)$. Therefore, $\gcd(f', s_2^{(p-1)/2}-1) = f'$. Next, the algorithm considers s_2+1. After computing $\gcd(f', s_2+1)$ and finding this to be 1, the algorithm computes $\gcd(f', (s_2+1)^{(p-1)/2}-1)$. The value of this gcd is f_2, and so U becomes $\{f_1, f_2, f_3\}$—the complete factorization of f. To see why this last gcd is f_2, observe that 2 is a quadratic nonresidue mod 1061, whereas $20 \equiv 284^2 \pmod{1061}$, and so $(s_2+1)^{(p-1)/2}-1$ corresponds to $(-2,0,-2)$. Therefore, $\gcd(f', (s_2+1)^{(p-1)/2}-1) = f_2$. □

In the next chapter, we will use the following result, which we obtain with a slight modification of our general factoring algorithm.

Theorem 2.5. *Let g be the product of k distinct monic irreducible polynomials of degree d. Then a single irreducible factor of g can be extracted deterministically using $O((\log p)dm^{1+\epsilon} + p^{1/2}(\log p)^2 m^{1+\epsilon})$ $\mathbf{F}_p$-operations, where $m = kd$.*

Proof. The idea is to simply redefine the procedure *Refine* so that *Refine*$(\{u\}, v)$ returns $\{u'\}$, where u' is defined as follows: if $\gcd(u,v)$ is a nontrivial divisor of u, then u' is the polynomial of smaller degree among $\gcd(u,v)$ and $u/\gcd(u,v)$; otherwise, $u' = u$. The termination condition of the outer loop of the refinement procedure (line 2 in Figure 2.1) needs to be modified so that it terminates when $U = \{u\}$ and $\deg u = d$. In the notation of the proof of the previous theorem, the number of i $(1 \le i \le d)$ for which $M_i > 0$ is at most $\log k$. □

2.4 Average Case Analysis

This section is devoted to a proof of the following theorem, which is concerned with the average-case behavior of the algorithm presented in the previous section.

Theorem 2.6. *Assume the input to our algorithm is drawn from a uniform distribution on all monic polynomials of degree n over $\mathbf{F}_p$. The probability that our algorithm fails to halt after $O((\log p)n^{2+\epsilon} + (\log p)^2 n^{1+\epsilon})$ $\mathbf{F}_p$-operations is at most $O((\log n \cdot \log p)^2/p)$. Furthermore, the average-case running time of our algorithm is $O((\log p)n^{2+\epsilon})$.*

More precisely, the first statement of the theorem says that there is a function $\tau(p,n) = O((\log p)n^{2+\epsilon} + (\log p)^2 n^{1+\epsilon})$ such that the probability that the number of $\mathbf{F}_p$-operations executed by our algorithm exceeds $\tau(n,p)$ is $O((\log n \cdot \log p)^2/p)$. Clearly, this theorem says something of interest only when p is large, and so in particular, we will assume that p is odd throughout this section.

Let f be a polynomial chosen at random from a uniform distribution on all monic polynomials of degree n over $\mathbf{F}_p$. We can partition the polynomials of degree n in $\mathbf{F}_p[X]$ according to their "factorization pattern." The factorization pattern π of f is an n-tuple $(k_1,\ldots,k_n)$ where k_d is the number of irreducible factors (counting multiplicities) of degree d that divide f. For example, if $f = X^2(X+1)(X^2+X+1)$, then $\pi = (3,1,0,0,0)$. We shall let r denote the number of irreducible factors of f, i.e. $r = k_1 + \cdots + k_n$.

Consider the action taken by our algorithm in factoring $f^{(d)}$ for some $1 \le d \le n$. Let K_d be the *total* number of times the body of the inner loop of the refinement procedure (lines 5–7 of Figure 2.1) is executed in factoring $f^{(d)}$.

The theorem will be a consequence of the following four lemmas.

Lemma 2.7. *Let p be an odd prime, let χ be the quadratic character on $\mathbf{F}_p$, and let t be an integer, with $0 < t < p$. Then if a and b are chosen at random from $\mathbf{F}_p$, the probability that $\chi(a+\delta) = \chi(b+\delta)$ for $\delta = 0,\ldots,t-1$ is $O(t^2/p + 1/2^t)$.*

Proof. Let J be the number of pairs $(a,b) \in \mathbf{F}_p^2$ such that $\chi(a+\delta) = \chi(b+\delta)$ for $\delta = 0,\ldots,t-1$. Then J is no more than t plus the number J' of pairs (a,b) such that $\chi((a+\delta)(b+\delta)) = 1$ for $\delta = 0,\ldots,t-1$. Now, J' is the number of pairs (a,b) for which there exist nonzero $c_1,\ldots,c_t$ in $\mathbf{F}_p$ such that

$$\begin{aligned} ab &= c_1^2 \\ (a+1)(b+1) &= c_2^2 \\ &\vdots \\ (a+t-1)(b+t-1) &= c_t^2. \end{aligned} \qquad (*)$$

Let N be the number of solutions $(a,b,c_1,\ldots,c_t) \in \mathbf{F}_p^{t+2}$ to $(*)$. We want to get a good upper bound on N. We have

$$\begin{aligned} N &= \sum_{a,b\in\mathbf{F}_p} (1+\chi(ab))\cdots(1+\chi((a+t-1)(b+t-1))) \\ &= \sum_{0\le e_1,\ldots,e_t\le 1} \sum_{a,b\in\mathbf{F}_p} \chi(a^{e_1}b^{e_1}\cdots(a+t-1)^{e_t}(b+t-1)^{e_t}) \\ &= \sum_{0\le e_1,\ldots,e_t\le 1} \sum_{a\in\mathbf{F}_p}\chi(a^{e_1}\cdots(a+t-1)^{e_t}) \sum_{b\in\mathbf{F}_p}\chi(b^{e_1}\cdots(b+t-1)^{e_t}). \end{aligned}$$

In this last expression, the term corresponding to $e_1 = \cdots = e_t = 0$ is p^2. We can use Lemma 2.2 to bound the magnitude of each of the other terms, obtaining

$$\begin{aligned} N &\le p^2 + p\sum_{l=1}^{t}\binom{t}{l}(l-1)^2 \\ &= p^2 + p\left(t(t-1)2^{t-2} - t2^{t-1} + 2^t - 1\right). \end{aligned}$$

We divide this quantity by 2^t to obtain a bound on the number of (a,b) for which there exist nonzero $c_1,\ldots,c_t$ satisfying $(*)$. Using the fact that $J \le t + J'$, we have

$$J \le p\left(\frac{t}{p} + \frac{p}{2^t} + \frac{t(t-1)}{4} - \frac{t}{2} + 1 - \frac{1}{2^t}\right).$$

Dividing the quantity on the right-hand side of this inequality by p^2 gives us the desired bound on the probability in question. □

Lemma 2.8. *Let $\pi = (k_1,\ldots,k_n)$ be a factorization pattern. Let t be an integer, with $0 < t \le \lceil \log p\rceil$. Then $\Pr(K_d > t|\pi) = O((k_dt)^2/2^t)$ for all $1 \le d \le n$.*

Proof. Let's fix the factorization pattern of f to be π. Suppose that $g = f^{(d)} = g_1 \ldots g_k$ (note that $k \le k_d$). Fix $\mathbf{F}_{p^d}$ to be some finite field with p^d elements. Let T be the trace from $\mathbf{F}_{p^d}$ down to $\mathbf{F}_p$. Let α and β be randomly chosen elements from $\mathbf{F}_{p^d}$.

Claim 1. $\Pr(K_d > t|\pi)$ *is no more than $k_d(k_d-1)/2$ times the conditional probability that $\chi(T(\alpha)+\delta) = \chi(T(\beta)+\delta)$ for $\delta = 0,\ldots,t-1$ given that*

$[\mathbf{F}_p(\alpha) : \mathbf{F}_p] = d$ *and* $[\mathbf{F}_p(\beta) : \mathbf{F}_p] = d$. To see this, first observe that $K_d > t$ implies that for some pair $1 \le i < j \le k$, we have $\chi(\theta_i(s_1) + \delta) = \chi(\theta_j(s_1)+\delta)$ for $\delta = 0, \ldots, t-1$. Next, observe that $\theta_i(s_1) = -T(x_i)$, where x_i is any root of g_i. Finally, observe that the conditional probability distribution of g given π is the same as that of a polynomial obtained by choosing k_d elements from a uniform distribution on all elements in $\mathbf{F}_{p^d}$ of degree d over $\mathbf{F}_p$, computing their minimum polynomials, and multiplying together the distinct polynomials among these. The claim follows immediately from these observations.

Claim 2. The probability that a randomly chosen element in $\mathbf{F}_{p^d}$ *has degree* d *over* $\mathbf{F}_p$ *is at least* $1/2$. To see this, let N_d be the number of elements of degree d over $\mathbf{F}_p$ in $\mathbf{F}_{p^d}$. Then we know that

$$\begin{aligned} N_d &\ge p^d - \sum_{\substack{e \mid d \\ e \neq d}} p^e \\ &\ge p^d - \sum_{e=1}^{\lfloor d/2 \rfloor} p^e \\ &\ge p^d - \left(\frac{p}{p-1}\right) p^{\lfloor d/2 \rfloor}. \end{aligned}$$

Dividing this quantity by p^d, we see that the probability that a randomly chosen element in $\mathbf{F}_{p^d}$ has degree d over $\mathbf{F}_p$ is at least

$$1 - \left(\frac{p}{p-1}\right) p^{-\lceil d/2 \rceil} \ge 1 - \frac{1}{p-1} \ge 1/2,$$

the last inequality following from the assumption that $p \ge 3$. This proves the claim.

By Claim 1, it will suffice to bound the conditional probability that $\chi(T(\alpha) + \delta) = \chi(T(\beta) + \delta)$ for $\delta = 0, \ldots, t-1$ given that $[\mathbf{F}_p(\alpha) : \mathbf{F}_p] = d$ and $[\mathbf{F}_p(\beta) : \mathbf{F}_p] = d$, which is at most

$$\frac{\Pr(\chi(T(\alpha) + \delta) = \chi(T(\beta) + \delta) \text{ for } \delta = 0, \ldots, t-1)}{\Pr([\mathbf{F}_p(\alpha) : \mathbf{F}_p] = d) \Pr([\mathbf{F}_p(\beta) : \mathbf{F}_p] = d)}.$$

Now, if α is a randomly chosen element of $\mathbf{F}_{p^d}$, then $T(\alpha)$ is just a randomly chosen element of $\mathbf{F}_p$, since T is a homomorphism from the additive group of

$\mathbf{F}_{p^d}$ onto the additive group of $\mathbf{F}_p$. By the previous lemma, and the hypothesis that $t \leq \lceil \log p \rceil$, the numerator in the above fraction is $O(t^2/2^t)$. By Claim 2, the denominator is at least $1/4$. The lemma now follows immediately. □

Lemma 2.9. *Let $\pi = (k_1, \ldots, k_n)$ be a factorization pattern. Then $E(K_d|\pi) = O(k_d^2 \min(d, k_d))$ for all $1 \leq d \leq n$.*

Proof. By definition, $E(K_d|\pi) = \sum_{t \geq 1} t \Pr(K_d = t|\pi)$. We can break this sum into two pieces: $1 \leq t \leq \lceil \log p \rceil$ and $t > \lceil \log p \rceil$.

Consider the first piece. By the previous lemma, we have

$$\begin{aligned} \sum_{1 \leq t \leq \lceil \log p \rceil} t \Pr(K_d = t|\pi) &\leq \sum_{1 \leq t \leq \lceil \log p \rceil} \Pr(K_d \geq t|\pi) \\ &= O\Big(\sum_{t \geq 1} (k_d t)^2/2^t\Big) \\ &= O(k_d^2). \end{aligned}$$

Now consider the second piece. From the proof of Theorem 2.4, we know that $K_d = O(\min(d, k_d) p^{1/2} \log p)$. And so, by the previous lemma, we have

$$\begin{aligned} \sum_{t > \lceil \log p \rceil} t \Pr(K_d = t|\pi) &= O(\min(d, k_d) p^{1/2} (\log p) \Pr(K_d > \lceil \log p \rceil |\pi)) \\ &= O(\min(d, k_d) p^{1/2} (\log p) \cdot (k_d \log p)^2/p) \\ &= O(k_d^2 \min(d, k_d)). \end{aligned}$$

And so the lemma is proved. □

Lemma 2.10. *If f is chosen at random from a uniform distribution on all monic polynomials of degree n in $\mathbf{F}_p[X]$, and if r denotes the number of irreducible factors of f (counting multiplicities), then $E(r^2) = O((\log n)^2)$.*

We remark that the constant implicit "O" expression in this lemma is absolute (in particular, it does not depend on p).

Proof. This lemma is an extension of results given in problems 4 and 5 of section 4.6.2 of volume 2 of Knuth [24]. Our proof is based on the proof found there, which uses generating functions. The generating functions that we use should be viewed strictly as formal power series, and all of the operations we perform on them, e.g., differentiation and exponentiation, as formal operations. The justification of the ordinary properties of these operations can be found in [31].

Let $G_p(z) = \sum_{n\geq 1} a_{np} z^n$, where a_{np} is the number of monic irreducible polynomials of degree n over $\mathbf{F}_p$. An argument sketched by Knuth [24, p. 624] shows that

$$\sum_{j\geq 1} G_p(z^j)/j = \ln(1/(1-pz)).$$

Let a_{npr} be the number of monic polynomials of degree n in $\mathbf{F}_p$ with exactly r irreducible factors. Let $U_p(z,w) = \sum_{n,r\geq 0} a_{npr} z^n w^r$. Because $E(r^2) = O(E(r(r-1)))$, it will suffice to prove that the coefficient of z^n in

$$\frac{\partial^2}{\partial w^2} U_p(z/p, 1)$$

is $O((\log n)^2)$.

First, observe that $U_p(z,w) = \exp\left(\sum_{k\geq 1} G_p(z^k) w^k/k\right)$. To see this, note that

$$\begin{aligned} U_p(z,w) &= (1 + zw + (zw)^2 + \cdots)^{a_{1p}} (1 + z^2 w + (z^2 w)^2 + \cdots)^{a_{2p}} \cdots \\ &= (1/(1-zw))^{a_{1p}} (1/(1-z^2 w))^{a_{2p}} \cdots \end{aligned}$$

Taking logarithms, we have

$$\begin{aligned} \ln U_p(z,w) &= a_{1p} \ln(1/(1-zw)) + a_{2p} \ln(1/(1-z^2 w)) + \cdots \\ &= a_{1p} \sum_{k\geq 1} z^k w^k/k + a_{2p} \sum_{k\geq 1} (z^2)^k w^k/k + \cdots \\ &= \sum_{k\geq 1} G_p(z^k) w^k/k. \end{aligned}$$

Using this expression for $U_p(z,w)$, differentiating twice with respect to w, and evaluating at $z = z/p$ and $w = 1$, we obtain

$$\frac{\partial^2}{\partial w^2} U_p(z/p, 1) = \Big(\sum_{k\geq 2} G_p(z^k/p^k)(k-1)\Big)/(1-z) + \Big(\sum_{k\geq 1} G_p(z^k/p^k)\Big)^2/(1-z).$$

We claim that for any fixed $t \geq 0$, the coefficient of z^n in $\sum_{k\geq 1} G_p(z^k/p^k) k^t$ is $O(1/n)$. To prove this claim, notice that $\sum_{k\geq 1} G_p(z^k/p^k) k^t =$

$\sum_{k,l\geq 1} k^t a_{lp}(z/p)^{lk}$. Using the fact the number of monic irreducible polynomials over $\mathbf{F}_p$ of degree l is no more than p^l/l, we have that the coefficient of z^n is

$$\begin{aligned}\sum_{l|n}(n/l)^t a_{lp}p^{-n} &= (n^t/p^n)\sum_{l|n} a_{lp}/l^t \\ &\leq (n^t/p^n)\sum_{l|n} p^l/l^{t+1} \\ &= O((n^t/p^n)(p^n/n^{t+1})) \\ &= O(1/n).\end{aligned}$$

From the claim in the previous paragraph, it is apparent that the coefficient of z^n in

$$\Big(\sum_{k\geq 2} G_p(z^k/p^k)(k-1)\Big)/(1-z)$$

is

$$O\Big(\sum_{1\leq k\leq n} 1/k\Big) = O(\log n),$$

and that the coefficient of z^n in

$$\Big(\sum_{k\geq 1} G_p(z^k/p^k)\Big)^2/(1-z)$$

is

$$O\Big(\sum_{1\leq k\leq n}\sum_{\substack{i,j\geq 1\\ i+j=k}} 1/ij\Big) = O((\log n)^2).$$

The lemma is proved. □

We are now in a position to prove the theorem. The total number of $\mathbf{F}_p$-operations executed by our algorithm is

$$O\Big((\log p)n^{2+\epsilon} + \log p\sum_d K_d(dk_d)^{1+\epsilon}\Big).$$

First, consider the probability P that $K_d > \log p$ for some d. We have

$$P \;\leq\; \sum_d \Pr(K_d > \log p)$$

$$
\begin{aligned}
&= \sum_d \sum_\pi \Pr(K_d > \log p|\pi) \Pr(\pi) \\
&= O(((\log p)^2/p) \sum_\pi \sum_d k_d^2 \Pr(\pi)) \quad \text{(Lemma 2.8)} \\
&= O(((\log p)^2/p) \sum_\pi r^2 \Pr(\pi)) \\
&= O(((\log p)^2/p) E(r^2)) \\
&= O((\log n \cdot \log p)^2/p) \quad \text{(Lemma 2.10)}.
\end{aligned}
$$

The first statement of the theorem is now clear.

The expected number of $\mathbf{F}_p$-operations executed by our algorithm is

$$
O\Big((\log p)n^{2+\epsilon} + (\log p)n^{1+\epsilon} E(\sum_d K_d)\Big).
$$

We have

$$
\begin{aligned}
E(\sum_d K_d) &= \sum_d E(K_d) \\
&= \sum_d E(E(K_d|\pi)) \\
&= \sum_d \sum_\pi E(K_d|\pi) \Pr(\pi) \\
&= O(\sum_\pi \sum_d k_d^2 \min(d, k_d) \Pr(\pi)) \quad \text{(Lemma 2.9)} \\
&= O(n^{1/2} \sum_\pi r^2 \Pr(\pi)) \\
&= O(n^{1/2} E(r^2)) \\
&= O(n^{1/2} (\log n)^2) \quad \text{(Lemma 2.10)}.
\end{aligned}
$$

The second statement of the theorem is now immediate.

2.5 Open Questions

The reader may be curious as to how the techniques in this chapter generalize to the problem of factoring polynomials over finite extensions of $\mathbf{F}_p$. Suppose p is fixed, and consider the problem of factoring a polynomial of degree n over an extension E of degree ν over $\mathbf{F}_p$. The probabilistic algorithm of Ben-Or

[6] runs in time $O((n\nu)^{2+\epsilon})$. It is natural to ask if this same bound can be achieved deterministically.

The bottleneck is again the construction of a separating set. Methods for separating set construction based on linear algebra require $O(M(n\nu))$ $\mathbf{F}_p$-operations, where $M(n\nu)$ is the number of $\mathbf{F}_p$-operations required to multiply two $n\nu$ by $n\nu$ matrices. Unfortunately, the method used in this section for constructing a separating set does not easily generalize. The obvious thing to try is the following. Suppose that we want to factor $g = g_1 \dots g_k$, where the g_i's are distinct irreducible polynomials over E of degree d. Next, let $m = kd$, let $R = E[X]/(g)$, and let $x = X \bmod g$. Analogous to what we did in Section 2.3, we next compute the coefficients of $(Y-x)(Y-x^p)\cdots(Y-x^{p^\mu})$, where $\mu = \mathrm{lcm}(d, \nu)$, and hope that these coefficients form a separating set. However, they do not if for any distinct pair g_i, g_j, the roots of g_i and g_j are conjugate over $\mathbf{F}_p$. The extreme example of this situation is where g is an irreducible polynomial over $\mathbf{F}_p$, but splits over E. So we ask if there is another way of constructing a small separating set for g in time $O((m\nu)^{2+\epsilon})$.

It is also natural to ask if the $p^{1/2+\epsilon}$ bound in the worst-case analysis of our algorithm can be improved. Can Burgess' techniques be adapted to our situation? Can we obtain a better bound assuming the Extended Riemann Hypothesis?

Chapter 3

Constructing Irreducible Polynomials over Finite Fields

3.1 Introduction

In this chapter, we consider the problem of constructing irreducible polynomials over finite fields. Such polynomials are used to implement arithmetic in extension fields found in many applications, including coding theory [7], cryptography [16], multivariate polynomial factoring [46], and parallel polynomial arithmetic [18].

Let n a positive integer. Consider the deterministic complexity of finding an irreducible polynomial of degree n in $\mathbf{F}_p[X]$ where p is prime. For this problem, there is no known deterministic polynomial time algorithm, i.e., an algorithm that runs in time polynomial in n and $\log p$. However, in many applications p is small, and so an algorithm that ran in time polynomial in n and p would be of value. We present one here. Specifically, we present a deterministic algorithm that on input n and p generates an irreducible polynomial in $F_p[X]$ of degree n, and—ignoring powers of $\log n$ and $\log p$—runs in time $O(p^{1/2}n^4)$. Thus, if p is fixed, e.g., $p = 2$, then our algorithm runs in polynomial time.

Our approach to constructing irreducible polynomials is as follows. In Section 3.2, we show that if we are given certain nonresidues in extension fields of $\mathbf{F}_p$, then we can deterministically generate an irreducible polynomial over $\mathbf{F}_p$ of degree n in polynomial time. More precisely, we prove the following result:

Assume that for each prime q such that $q \mid n, q \neq p$, we are given a splitting field K of $X^q - 1$ over $\mathbf{F}_p$ and a q-th nonresidue in K. Then we can find an irreducible polynomial over F of degree n deterministically in time polynomial in n and $\log p$.

In Section 3.3, we go on to show that given an oracle for factoring polynomials over $\mathbf{F}_p$, these extension fields and nonresidues—and hence irreducible polynomials over $\mathbf{F}_p$ of degree n—can be constructed deterministically in polynomial time:

The problem of constructing an irreducible polynomial over $\mathbf{F}_p$ of degree n can be deterministically reduced in time bounded by a polynomial in n and $\log p$ to the problem of factoring polynomials over $\mathbf{F}_p$.

We obtain a deterministic algorithm for generating irreducible polynomials by replacing the oracle by any variant of Berlekamp's deterministic factoring algorithm. It turns out that factoring is the bottleneck in the resulting algorithm (in terms of both n and p), but with Theorem 2.5 we obtain the following result:

We can deterministically construct an irreducible polynomial over $\mathbf{F}_p$ of degree n with $O(p^{1/2}(\log p)^3 n^{3+\epsilon} + (\log p)^2 n^{4+\epsilon})$ $\mathbf{F}_p$-operations.

In particular, if p is a fixed prime, then we can deterministically construct an irreducible polynomial over $\mathbf{F}_p$ of degree n in time $O(n^{4+\epsilon})$.

In Sections 3.4 and 3.5, we prove some extensions and related results, and in Section 3.6, we conclude with some open questions. In Section 3.4, we show that our results on finding irreducible polynomials extend to nonprime finite fields:

Given an extension E over $\mathbf{F}_p$ of degree ν, we can construct an irreducible polynomial over E of degree n deterministically with $O(p^{1/2}(\log p)^3 n^{3+\epsilon} + (\log p)^2 n^{4+\epsilon} + (\log p) n^{4+\epsilon} \nu^{2+\epsilon})$ $\mathbf{F}_p$-operations.

This result allows us to deterministically construct in polynomial time an irreducible polynomial of specified degree over a finite field of small characteristic. If p is fixed, then our algorithm runs in time $O(n^{4+\epsilon}\nu^{2+\epsilon})$. The proof of this result reduces the problem of finding an irreducible polynomial over E of degree n to the problem of factoring polynomials over $\mathbf{F}_p$ via the problem of constructing, for each prime $q \mid n, q \neq p$, the splitting field of $X^q - 1$ over $\mathbf{F}_p$, along with a q-th nonresidue in this field, or just a primitive q-th root of unity if q also happens to divide ν.

In Section 3.5, we give another algorithm for constructing irreducible polynomials over $\mathbf{F}_p$ that runs in time polynomial in n and p. This method uses the results of Section 3.2, but constructs nonresidues using a technique completely different from that in Section 3.3. It makes use of an analogue of Ankeny's theorem on the least quadratic nonresidue modulo a prime [3] and of an analogue of Pratt's primality certificate [33]. Unlike the algorithm in Section 3.3, the algorithm in this section can be easily recast as a fast parallel algorithm when p is small. We prove:

The problem of constructing an irreducible polynomial over $\mathbf{F}_p$ of degree n, given p and n with p bounded by a polynomial in n, can be solved by a family of uniform Boolean circuits of size polynomial in n and depth polynomial in $\log n$.

Related Work

Even for small values of p, previous algorithms for generating irreducible polynomials over $\mathbf{F}_p$ suffer from at least one of three drawbacks: they rely on *a source of randomness*, they rely on *unproven conjectures* in the proofs of their running times, or they generate polynomials of degree only *approximately* n. Rabin [35] gives a probabilistic polynomial time algorithm. Adleman and Lenstra [1] give a deterministic algorithm that runs in polynomial time assuming the Extended Riemann Hypothesis (ERH). They also give a deterministic polynomial time algorithm that generates an irreducible polynomial of degree only approximately n. Von zur Gathen [44] gives several deterministic algorithms that are efficient in practice, but his proofs of their running times rely on unproven conjectures, and they generate irreducible polynomials of degree only approximately n.

We also mention two other results on constructing irreducible polynomials, of which our results were obtained independently. In a paper on factoring polynomials over finite fields, Evdokimov [19] gives another proof that irreducible polynomials of specified degree can be constructed deterministically in polynomial time assuming the ERH. Evdokimov's method of constructing irreducible polynomials is similar to ours in that Evdokimov essentially reduces this problem to the problem of finding various nonresidues in extension fields of $\mathbf{F}_p$; however, Evdokimov constructs these nonresidues by appealing to the ERH (making use of results in [21] and [25]), and does not address the problem of constructing these nonresidues deterministically without relying on the ERH.

In another paper, Varshamov [42] describes a method for constructing irreducible polynomials of specified degree; however, in some cases the method either breaks down or appears to require time greater than a polynomial in n and p. One problem with Varshamov's method is that it requires the calculation of the multiplicative order of elements in extensions of $\mathbf{F}_p$, for which no polynomial time algorithm is known (even for small p).

3.2 Reduction to Constructing Cyclotomic Extensions and Finding Nonresidues

This section is devoted to a proof of the following result.

Theorem 3.1. *Assume that for each prime q such that $q \mid n, q \neq p$, we are given a splitting field K of $X^q - 1$ over $\mathbf{F}_p$ and a q-th nonresidue in K. Then we can find an irreducible polynomial over $\mathbf{F}_p$ of degree n deterministically in time polynomial in n and $\log p$.*

The splitting field of $X^q - 1$ is the smallest extension of $\mathbf{F}_p$ containing a primitive q-th root of unity. It is also the smallest extension of $\mathbf{F}_p$ containing q-th nonresidues. From group theory, we see that this is just $\mathbf{F}_{p^m}$ where m is the smallest positive integer such that q divides $p^m - 1$, the order of the group $\mathbf{F}_{p^m}^*$. That is, m is the order of $p \bmod q$. Note that $m \mid q-1$. The hypothesis of Theorem 3.1 means that we are given an irreducible polynomial f over $\mathbf{F}_p$ of degree m and a q-th nonresidue a in $\mathbf{F}_p(\alpha)$ where α is a root of f.

We now describe our algorithm. Let $n = q_1^{e_1} \cdots q_r^{e_r}$ be the prime factorization of n. We first construct irreducible polynomials over $\mathbf{F}_p$ of degree $q_i^{e_i}$ for $i = 1, \ldots, r$. We then "combine" these polynomials to form an irreducible polynomial of degree n.

Step 1: Constructing Irreducible Polynomials of Prime Power Degree

Let $1 \leq i \leq r$ be fixed, and let $q = q_i, e = e_i$. We want to construct an irreducible polynomial in $\mathbf{F}_p[X]$ of degree q^e. We break the problem down into three cases: (1) $q \neq 2, \neq p$, (2) $q = 2, \neq p$, and (3) $q = p$.

Case 1: $q \neq 2, \neq p$

Let m be the order of p mod q. By hypothesis, we are given an irreducible polynomial f of degree m over $\mathbf{F}_p$, and a q-th nonresidue a in $K = \mathbf{F}_p(\alpha)$ where α is a root of f.

We will make use of the following result, which is proved in [26, p. 331].

Lemma 3.2. *Let K be a field and d an integer ≥ 2. Let $a \in K, a \neq 0$. Assume that for all prime numbers t dividing d, a is a t-th nonresidue in K, and if $4 \mid d$ then a is not equal to -4 times a t-th residue. Then $X^d - a$ is irreducible in $K[X]$.*

By Lemma 3.2, the polynomial $X^{q^e} - a \in K[X]$ is irreducible. We can represent the field $E = \mathbf{F}_{p^{mq^e}}$ by $K(\beta)$, where β is a root of $X^{q^e} - a$. Now, $H = \mathbf{F}_{p^{q^e}}$ is a subfield of E. We have the following picture:

$$\begin{tikzcd} & E = K(\beta) \arrow[dl, dash, "q^e"'] \arrow[dr, dash, "m"] & \\ K = \mathbf{F}_p(\alpha) \arrow[dr, dash, "m"'] & & H \arrow[dl, dash, "q^e"] \\ & \mathbf{F}_p & \end{tikzcd}$$

It will suffice to find an element γ in E of degree q^e over $\mathbf{F}_p$. We can then construct its minimum polynomial over $\mathbf{F}_p$ by computing $(X - \gamma)(X - \gamma^p) \cdots (X - \gamma^{p^{q^e-1}})$. This will be an irreducible polynomial of degree q^e over $\mathbf{F}_p$. But finding an element in E of degree q^e over $\mathbf{F}_p$ is easy. Let T be the trace from E down to H; that is, for any x in E, $T(x) = x + x^\sigma + \cdots + x^{\sigma^{m-1}}$, where σ is the generator of the Galois group of E over H given by $x \mapsto x^{p^{q^e}}$. Then we claim that $\gamma = T(\beta)$ has degree q^e over $\mathbf{F}_p$.

To prove this claim, suppose to the contrary that γ has degree q^t over $\mathbf{F}_p$, where $t < e$. Then it is easy to see that γ has degree q^t over K. Now, $[K(\beta) : K(\beta^q)] = q$, and so in particular, γ lies in $K(\beta^q)$. For $i = 0, \ldots, m-1$, let $p^{iq^e} = x_i q + y_i$, where $0 < y_i < q$. It is easily seen that the y_i's are distinct, since p (which is $\equiv p^{q^e} \pmod{q}$) has order m mod q. This gives us an equation

$$(\beta^q)^{x_0}\beta^{y_0} + \cdots + (\beta^q)^{x_{m-1}}\beta^{y_{m-1}} - \gamma = 0.$$

Thus, β is a root of a nonzero polynomial over $K(\beta^q)$ of degree less than q. But this contradicts the fact that β has degree q over $K(\beta^q)$, and so the claim is proved.

Case 2: $q = 2, \neq p$

We want to find an irreducible polynomial of degree 2^e. In this case, as in case 1, we make use of Lemma 3.2. Since p is odd, $p \equiv \pm 1 \pmod{4}$. Suppose $p \equiv 1 \pmod{4}$. Then $(-1)^{(p-1)/2} = 1$, and so -1 has a square root in $\mathbf{F}_p$. Therefore, if we have an element $a \in \mathbf{F}_p$ that is not a square, then we certainly cannot have $a = -4b^4$, since $-4b^4$ is a square. Thus, the hypotheses of Lemma 3.2 are already satisfied, and so $X^{2^e} - a$ is irreducible.

Now suppose $p \equiv -1 \pmod{4}$. In this case, we can quickly find an irreducible polynomial of degree 2^e deterministically. We have $(-1)^{(p-1)/2} = -1$, so -1 does not have a square root in $\mathbf{F}_p$, and therefore $X^2 + 1$ is irreducible. If $e = 1$, we are done. Otherwise, we can proceed as follows. We construct the field $\mathbf{F}_p(i)$ where $i = \sqrt{-1}$. Since -1 has a square root in $\mathbf{F}_p(i)$, if we find an $a \in \mathbf{F}_p(i)$ that is not a square, then $X^{2^{e-1}} - a$ is an irreducible polynomial in $\mathbf{F}_p(i)[X]$ (by reasoning identical to that in the previous paragraph). Let $E = \mathbf{F}_p(i, \alpha)$ where α is a root of $X^{2^{e-1}} - a$. It is easy to see that $E = \mathbf{F}_p(\alpha)$ (since $[\mathbf{F}_p(i, \alpha) : \mathbf{F}_p] = \mathrm{lcm}([\mathbf{F}_p(i) : \mathbf{F}_p], [\mathbf{F}_p(\alpha) : \mathbf{F}_p])$, and so it will suffice to compute the minimum polynomial of α over $\mathbf{F}_p$, which has degree 2^e. Let σ be the automorphism on $\mathbf{F}_p(i)$ defined by $i \mapsto -i$. Then the minimum polynomial for α over $\mathbf{F}_p$ is just $(X^{2^{e-1}} - a)(X^{2^{e-1}} - a^\sigma)$.

So we have reduced the problem to finding a quadratic nonresidue in $\mathbf{F}_p(i)$. This is easily done as follows. $\mathbf{F}_p(i)^*$ is a cyclic group of order $p^2 - 1$. Write $p^2 - 1 = l2^k$, l odd. If we take $k - 2$ successive square roots of i, we will obtain a primitive 2^k-th root of unity in $\mathbf{F}_p(i)$. This must be a quadratic nonresidue; otherwise, its square root would be an element of order 2^{k+1} in $\mathbf{F}_p(i)^*$, which is impossible by Lagrange's theorem.

So we have reduced the problem to taking square roots in $\mathbf{F}_p(i)$. But this can easily be done using the formula

$$\sqrt{\alpha} = (1 + \alpha^{(p-1)/2})^{(p-1)/2} \cdot \alpha^{(p+1)/4},$$

which holds for every quadratic residue α in $\mathbf{F}_p(i)$, except if $\alpha^{(p-1)/2} = -1$, in which case $\sqrt{\alpha} = i\alpha^{(p+1)/4}$.

Case 3: $q = p$

We want to construct an irreducible polynomial of degree p^e. Our approach in this case follows that of Adleman and Lenstra [1]. We will show how to inductively construct a sequence of irreducible polynomials $f_1, f_2, \ldots, f_e$ over $\mathbf{F}_p$ of degrees $p, p^2, \ldots, p^e$.

Lemma 3.3. *The polynomial $X^p - X - 1$ is irreducible in $\mathbf{F}_p[X]$. Furthermore, if K is an extension of $\mathbf{F}_p$, and $a \in K$, and the polynomial $X^p - X - a$ is irreducible in $K[X]$, and $E = K(\alpha)$ where α is a root of $X^p - X - a$, then the polynomial $X^p - X - a\alpha^{p-1}$ is irreducible in $E[X]$.*

Proof. According to Artin-Schreier theory (see, e.g., [26, p. 325]), over any field L of characteristic p, the polynomial $X^p - X - c$ (where $c \in L$) is either irreducible or splits completely. The first statement of the lemma follows immediately from this. To prove the second statement, suppose that $X^p - X - a\alpha^{p-1}$ is not irreducible in $E[X]$. Then it has a root β in E, which we can write as $\beta = \sum_{i=0}^{p-1} b_i\alpha^i$ where the b_i's are in K. Substituting this expression for β into the equation $\beta^p - \beta - a\alpha^{p-1} = 0$, and replacing α^p by $\alpha + a$, we obtain an equation

$$\sum_{i=0}^{p-1} b_i^p(\alpha + a)^i - \sum_{i=0}^{p-1} b_i\alpha^i - a\alpha^{p-1} = 0.$$

In the expansion of the left hand side of this equation, the coefficient of α^{p-1} is $b_{p-1}^p - b_{p-1} - a$, which is nonzero by virtue of the fact that $X^p - X - a$ is irreducible over K. Thus, α is a root a polynomial of degree $p - 1$ over K, which is impossible, and so the lemma is proved. □

Let $f_1 = X^p - X - 1$. By the previous lemma, this is irreducible. Suppose that we have computed f_t, $t \geq 1$. Let $K = \mathbf{F}_p(\alpha)$ where α is a root of f_t.

If $t = 1$, let $a = \alpha^{p-1}$; otherwise, let $a = \alpha^{2p-1} - \alpha^p$. By the previous lemma, the polynomial $X^p - X - a$ is irreducible over K. Let $E = K(\beta)$, where β is a root of $X^p - X - a$. It is easy to see that $E = \mathbf{F}_p(\beta)$ (since $[\mathbf{F}_p(\alpha, \beta) : \mathbf{F}_p] = \mathrm{lcm}([\mathbf{F}_p(\alpha) : \mathbf{F}_p], [\mathbf{F}_p(\beta) : \mathbf{F}_p])$. Let f_{t+1} be the minimum polynomial of β over $\mathbf{F}_p$, which we compute as $\prod_{i=0}^{p^t-1}(X^p - X - a^{p^i})$. Observe that the polynomial $X^p - X - a\beta^{p-1} = X^p - X - (\beta^{2p-1} - \beta^p)$ is irreducible over E.

Step 2: "Combining" Irreducible Polynomials of Prime Power Degree

Suppose we have constructed irreducible polynomials over $\mathbf{F}_p$ of degrees $q_1^{e_1}, \ldots, q_r^{e_r}$. We will show how to inductively construct a sequence of irreducible polynomials over $\mathbf{F}_p$ of degrees $q_1^{e_1}, q_1^{e_1} q_2^{e_2}, \ldots, q_1^{e_1} \cdots q_r^{e_r} = n$. It will suffice to solve the following problem: given two irreducible polynomials $f, g \in \mathbf{F}_p[X]$ of degrees a and b, where $\gcd(a, b) = 1$, find an irreducible polynomial of degree ab.

Lemma 3.4. *Let α and β be elements in the the algebraic closure of $\mathbf{F}_p$. Suppose that $[\mathbf{F}_p(\alpha) : \mathbf{F}_p] = a$, $[\mathbf{F}_p(\beta) : \mathbf{F}_p] = b$, and $\gcd(a, b) = 1$. Then $[\mathbf{F}_p(\alpha, \beta) : \mathbf{F}_p] = [\mathbf{F}_p(\alpha + \beta) : \mathbf{F}_p] = ab$.*

Proof. We have $[\mathbf{F}_p(\alpha, \beta) : \mathbf{F}_p] = \mathrm{lcm}([\mathbf{F}_p(\alpha) : \mathbf{F}_p], [\mathbf{F}_p(\beta) : \mathbf{F}_p]) = ab$. Any maximal proper subfield of $\mathbf{F}_p(\alpha, \beta)$ (i.e., $\mathbf{F}_{p^{ab/r}}$ where r is prime) must contain either α or β, but not both, and hence cannot contain $\alpha + \beta$. Therefore, $\mathbf{F}_p(\alpha + \beta) = \mathbf{F}_p(\alpha, \beta)$. □

Suppose that f and g are given as described above. Then this lemma allows us to construct a tower of fields $\mathbf{F}_p \subset \mathbf{F}_p(\alpha) \subset \mathbf{F}_p(\alpha, \beta)$ where α is a root of f and β is a root of g. The degree of the first step in the tower is a and the degree of the second is b. We can construct the minimal polynomial of $\alpha + \beta$ over $\mathbf{F}_p$ by computing $(X - (\alpha + \beta))(X - (\alpha + \beta)^p) \cdots (X - (\alpha + \beta)^{p^{ab-1}})$. This is an irreducible polynomial over $\mathbf{F}_p$ of degree ab.

Complexity Analysis

The following breakdown of the complexity of our algorithm, in terms of $\mathbf{F}_p$-operations, is easily obtained by a straightforward application of Lemma 1.7

to the computations performed by our algorithm:

Step 1	Case 1	$O((\log p)(mq^e)^{2+\epsilon})$
	Case 2	$O(2^e + (\log p)^2)$
	Case 3	$O((p^{2e-1})^{1+\epsilon})$
Step 2		$O((\log p)n^{2+\epsilon})$

3.3 Reduction to Factoring

In this section, we give a deterministic reduction from finding irreducible polynomials over $\mathbf{F}_p$ to the problem of factoring polynomials over $\mathbf{F}_p$. Then, using this reduction in conjunction with the factoring algorithm in Chapter 2, we obtain a deterministic algorithm for finding irreducible polynomials.

Theorem 3.5. *The problem of constructing an irreducible polynomial over $\mathbf{F}_p$ of degree n can be deterministically reduced in time bounded by a polynomial in n and $\log p$ to the problem of factoring polynomials over $\mathbf{F}_p$.*

Proof. Let q be a prime, $q \mid n, q \neq p$. Let m be the order of p mod q. By Theorem 3.1, it will suffice to find, for each such q, an irreducible polynomial f of degree m, and a q-th nonresidue in $\mathbf{F}_p(\alpha)$ where α is a root of f.

The basic idea is to factor the cyclotomic polynomial $\Phi_q = X^{q-1} + \cdots + 1$, obtaining an irreducible polynomial of degree m. This gives us $\mathbf{F}_{p^m}$ and a primitive q-th root of unity ξ in $\mathbf{F}_{p^m}$. Now, $\mathbf{F}_{p^m}^*$ is a cyclic group of order $p^m - 1$. Suppose that $p^m - 1 = lq^k$ where $\gcd(l, q) = 1$. Then if we take $k-1$ successive q-th roots of ξ, we obtain a primitive q^k-th root of unity in $\mathbf{F}_{p^m}$. This must be a q-th nonresidue; otherwise, its q-th root would be an element of order q^{k+1} in $\mathbf{F}_{p^m}^*$, which is impossible by Lagrange's theorem. So we have reduced the problem to finding roots of polynomials of the form $X^q - c$ over $\mathbf{F}_{p^m}$. Berlekamp [8] gives a reduction from factoring in $\mathbf{F}_{p^m}[X]$ to factoring in $\mathbf{F}_p[X]$. We give an explicit construction, tailoring Berlekamp's reduction to our particular application.

We inductively construct a sequence of irreducible polynomials $f^{(1)}, \ldots, f^{(k)}$ in $\mathbf{F}_p[X]$ of degree m where the roots of $f^{(i)}$ are primitive q^i-th roots of unity. We let $f^{(1)}$ be any irreducible factor of Φ_q. It is clear that the roots of $f^{(1)}$ are primitive q-th roots of unity. For $i = 2, \ldots, k$, we let $f^{(i)}$ be any irreducible factor of $f^{(i-1)}(X^q)$. Since the roots of $f^{(i-1)}$ are primitive q^{i-1}-th roots of unity, the roots of $f^{(i)}$ must be primitive q^i-th roots of unity.

Computing the sequence $f^{(1)}, \ldots, f^{(k)}$ requires us to factor one polynomial of degree $q-1$ and $k-1 < m \log p$ polynomials of degree mq. Each of these polynomials is the product of distinct irreducible polynomials of degree m. We then put $f = f^{(k)}$. Any root α of f is a q-th nonresidue in $\mathbf{F}_p(\alpha)$. The theorem is proved. $\square$

Using the algorithm of Theorem 3.5, in conjunction with Theorem 2.5 and the complexity analysis of Section 3.2, we immediately obtain the following theorem.

Theorem 3.6. *We can deterministically construct an irreducible polynomial over* $\mathbf{F}_p$ *of degree* n *with* $O(p^{1/2}(\log p)^3 n^{3+\epsilon} + (\log p)^2 n^{4+\epsilon})$ $\mathbf{F}_p$*-operations.*

Example. We will illustrate the workings of our algorithm with an example. Let $p = 7$ and suppose we want to construct an irreducible polynomial of degree $q = 5$ over $\mathbf{F}_p$. Let m be the order of p mod q, i.e. m is the smallest positive integer such that $p^m \equiv 1 \pmod{q}$. In this case, m turns out to be 4. Therefore, the polynomial $\Phi_q = X^4 + X^3 + X^2 + X + 1$ is irreducible over $\mathbf{F}_p$, and we set f_1 equal to Φ_q. Now, $p^m \equiv 1 \pmod{q^2}$, and so the roots of f_1 are not q-th nonresidues in $\mathbf{F}_{p^m}$. Therefore, we need to factor $f_1(X^q) = X^{20} + X^{15} + X^{10} + X^5 + 1$. Factoring $f_1(X^q)$, we find that one of its irreducible factors is $f_2 = X^4 + 6X^3 + 5X^2 + 6X + 1$. Since $p^m \not\equiv 1 \pmod{q^3}$, we can deduce that the roots of f_2 are q-th nonresidues in $\mathbf{F}_{p^m}$.

As described, our algorithm would construct the field $\mathbf{F}_p(\alpha)$, where α is a root of f_2, and then extend this field by adjoining to it a root β of $X^q - \alpha$. However, we can simplify matters by taking β to be a root of the irreducible polynomial $f_2(X^q) = X^{20} + 6X^{15} + 5X^{10} + 6X^5 + 1$. The algorithm computes

$$\gamma = T(\beta) = \beta + \beta^{p^q} + \beta^{p^{2q}} + \beta^{p^{3q}}.$$

Performing this calculation, we find that

$$\gamma = 6\beta^{19} + 6\beta^{18} + \beta^{14} + 2\beta^{13} + 4\beta^{12} + 2\beta^9 + 4\beta^8 + 2\beta^7 + \beta^4 + 3\beta^3 + \beta^2 + \beta.$$

The algorithm next computes the coefficients of

$$h(Y) = (Y - \gamma)(Y - \gamma^p)(Y - \gamma^{p^2})(Y - \gamma^{p^3})(Y - \gamma^{p^4}),$$

obtaining

$$h(Y) = Y^5 + 4Y^3 + 3Y^2 + 2Y + 6,$$

which is an irreducible polynomial over $\mathbf{F}_p$. $\square$

3.4 Irreducible Polynomials over Extension Fields

In this section, we describe an algorithm for constructing irreducible polynomials over finite extensions of $\mathbf{F}_p$, proving the following theorem.

Theorem 3.7. *Given an extension E over $\mathbf{F}_p$ of degree ν, we can construct an irreducible polynomial over E of degree n deterministically with $O(p^{1/2}(\log p)^3 n^{3+\epsilon} + (\log p)^2 n^{4+\epsilon} + (\log p) n^{4+\epsilon}\nu^{2+\epsilon})$ $\mathbf{F}_p$-operations.*

The hypothesis of this theorem means that $E = \mathbf{F}_p(\theta)$ where θ is a root of a given irreducible polynomial over $\mathbf{F}_p$ of degree ν. The algorithm described in Section 3.3 could be adapted to this situation with a few straightforward modifications. However, we will describe a slightly more complicated, but more efficient, algorithm.

Implicit in our algorithm is a reduction to the problem of factoring polynomials over $\mathbf{F}_p$ via the problem of constructing, for each prime $q \mid n, q \neq p$, the splitting field of $X^q - 1$ over $\mathbf{F}_p$, along with a q-th nonresidue in this field, or just a primitive q-th root of unity if q also happens to divide ν.

A straightforward implementation of our algorithm, making use of the running time bounds in Lemma 1.7 for performing polynomial arithmetic and the factoring algorithm of Theorem 2.5, will achieve the stated running time bound.

We now describe our algorithm. As in the proof of Theorem 3.1, it will suffice to construct irreducible polynomials over E of prime power degree q^e, and then combine these to obtain an irreducible polynomial over E of degree n. We consider two possibilities: either q does not divide ν, or it does.

In the first case, it will suffice to construct an irreducible polynomial over $\mathbf{F}_p$ of degree q^e, for this polynomial will remain irreducible over the larger field E. The algorithm in Section 3.3 can be used for this.

In the second case, let $\nu = q^k l$ where $\gcd(q, l) = 1$. It will then suffice to find a polynomial of degree q^e over $\mathbf{F}_p(\vartheta)$, where ϑ is an element in E of degree

q^k over $\mathbf{F}_p$, for this polynomial will remain irreducible over E. To facilitate efficient computation in $\mathbf{F}_p(\vartheta)$, we construct the minimum polynomial of ϑ over $\mathbf{F}_p$ by computing $(X-\vartheta)(X-\vartheta^p)\cdots(X-\vartheta^{p^{q^k-1}})$. We can find such an element ϑ quickly in the following way. Construct the minimum polynomial of θ over $\mathbf{F}_{p^{q^k}}$ by computing $(X-\theta)(X-\theta^\sigma)\cdots(X-\theta^{\sigma^{l-1}})$, where σ generates the Galois group of E over $\mathbf{F}_{p^{q^k}}$. Suppose this polynomial is $\vartheta_0+\vartheta_1X+\cdots+\vartheta_{l-1}X^{l-1}+X^l$. Then it is easy to see that $[\mathbf{F}_p(\vartheta_0,\ldots,\vartheta_{l-1}):\mathbf{F}_p]=q^k$, and so one of the ϑ_i's must have degree q^k over $\mathbf{F}_p$. We can examine each ϑ_i in turn until we find one such that $\vartheta_i^{p^{q^{k-1}}}\neq\vartheta_i$. Now let $\vartheta=\vartheta_i$. This has degree q^k over $\mathbf{F}_p$.

To construct an irreducible polynomial of degree q^e over $\mathbf{F}_p(\vartheta)$, as in Section 3.2, we break the problem into three cases: (1) $q\neq 2,\neq p$, (2) $q=2,\neq p$, and (3) $q=p$.

Case 1: $q\neq 2,\neq p$

We assume that we have the field $\mathbf{F}_p(\xi)$, where ξ is a primitive q-th root of unity, which we can obtain by factoring the q-th cyclotomic polynomial. Let $m=[\mathbf{F}_p(\xi):\mathbf{F}_p]$, i.e., m is the order of p mod q. Since m and q are relatively prime, we can construct the tower of fields $\mathbf{F}_p\subset\mathbf{F}_p(\vartheta)\subset\mathbf{F}_p(\vartheta,\xi)$ where the degree of the first step in the tower is q^k and the degree of the second is m.

We proceed to find a q-th nonresidue a in $\mathbf{F}_p(\vartheta,\xi)$ as follows. Let L be the subfield $\mathbf{F}_{p^{mq^{k-1}}}$ of $\mathbf{F}_p(\vartheta,\xi)$, and let σ generate the Galois group of $\mathbf{F}_p(\vartheta,\xi)$ over L. We compute the Lagrange resolvents

$$(\vartheta^i)+\xi(\vartheta^i)^\sigma+\cdots+\xi^{q-1}(\vartheta^i)^{\sigma^{q-1}}$$

for $i=1,\ldots,q-1$. One can show that one of these resolvents, call it a, must be nonzero, and that a^q is a q-th nonresidue in L (see page 179 of [41]). It follows easily from Lemma 3.2 that a is a q-th nonresidue in $\mathbf{F}_p(\vartheta,\xi)$, and that the polynomial $X^{q^e}-a$ is irreducible over $\mathbf{F}_p(\vartheta,\xi)$. So if we adjoin a root β of this polynomial to $\mathbf{F}_p(\vartheta,\xi)$, we obtain the tower of fields $\mathbf{F}_p\subset\mathbf{F}_p(\vartheta)\subset\mathbf{F}_p(\vartheta,\xi)\subset\mathbf{F}_p(\vartheta,\xi,\beta)$. We can now compute $\gamma=T(\beta)$ where T is the trace from $\mathbf{F}_p(\vartheta,\xi,\beta)$ down to $\mathbf{F}_{p^{q^{k+e}}}$. By reasoning identical to that in Section 3.2, we can prove that γ has degree q^e over $\mathbf{F}_p(\vartheta)$, and hence we can compute the minimum polynomial of γ over $\mathbf{F}_p(\vartheta)$ to obtain an irreducible polynomial of degree q^e over $\mathbf{F}_p(\vartheta)$.

Case 2: $q = 2, \neq p$

Let L be the subfield $\mathbf{F}_{p^{2^{k-1}}}$ of $\mathbf{F}_p(\vartheta)$, and let σ generate the Galois group of $\mathbf{F}_p(\vartheta)$ over L. Compute the Lagrange resolvent $a = \vartheta - \vartheta^\sigma$. Then a^2 is a quadratic nonresidue in L. If $k > 1$ or $p \equiv 1 \pmod 4$, then a is a quadratic nonresidue in $\mathbf{F}_p(\vartheta)$ and the polynomial $X^{2^e} - a$ is irreducible over $\mathbf{F}_p(\vartheta)$. Otherwise, $k = 1$ and $p \equiv -1 \pmod 4$. Let $p^2 - 1 = 2^s t$, and take $s - 2$ successive square roots of a in $\mathbf{F}_p(\vartheta)$ using the formula in Section 3.2, obtaining a 2^{s-2}-th root b of a. Then b is a quadratic nonresidue in $\mathbf{F}_p(\vartheta)$, and so $X^{2^e} - b$ is irreducible over $\mathbf{F}_p(\vartheta)$.

Case 3: $q = p$

As in Section 3.2, we inductively construct a sequence of irreducible polynomials over $\mathbf{F}_p(\vartheta)$ of degrees $p, p^2, \ldots, p^e$. Everything is essentially the same as in Section 3.2, but to get this inductive process started, we need to find an element a in $\mathbf{F}_p(\vartheta)$ such that $X^p - X - a$ is irreducible. A nice way to do this, given that we have the minimum polynomial of ϑ over $\mathbf{F}_p$, is as follows. Suppose the minimum polynomial of ϑ over $\mathbf{F}_p$ is $X^{p^k} + a_1 X^{p^k - 1} + \cdots + a_{p^k}$. Let i be the least positive integer smaller than p^k such that $i \not\equiv 0 \pmod p$ and $a_i \neq 0$. We claim that such an i exists, and that $X^p - X - \vartheta^i$ is irreducible over $\mathbf{F}_p(\vartheta)$.

To prove this claim, we first observe that such an i must exist, since otherwise the minimum polynomial of ϑ over $\mathbf{F}_p$ would be a perfect p-th power. Second, one can show that the polynomial $X^p - X - a$ is irreducible over $\mathbf{F}_p(\vartheta)$ if and only if $T(a) \neq 0$ where T is the trace from $\mathbf{F}_p(\vartheta)$ down to $\mathbf{F}_p$ (this follows from Hilbert's Theorem 90, and the Artin-Schreier Theorem on p. 325 of [26]). But using Newton's formulas (Lemma 2.1) we have the following recurrence relation:

$$T(\vartheta^i) + a_1 T(\vartheta^{i-1}) + \cdots + a_{i-1} T(\vartheta) + i a_i = 0 \qquad (i = 1, \ldots, p^k).$$

The claim is now immediate.

3.5 Another Algorithm for Finding Irreducible Polynomials

In this section, we describe another deterministic algorithm for constructing irreducible polynomials over $\mathbf{F}_p$ of degree n. This algorithm, like that of Section 3.3, works by finding appropriate nonresidues in extension fields of $\mathbf{F}_p$, and then uses the construction of Section 3.2 to obtain an irreducible polynomial; however, the method used to construct these nonresidues is different, and was suggested to the author by Lenstra [29].

The algorithm in this section has the advantage that it is somewhat simpler than the algorithm in Section 3.3. It also has the advantage that it is easily recast as a fast parallel algorithm when p is small, whereas the algorithm in Section 3.3 is not. It has the disadvantage that the dependence on p in the running time may be proportional to $p^{1+\epsilon}$, whereas the dependence on p in the running time of the algorithm in Section 3.3 is at worst proportional to $p^{1/2+\epsilon}$.

We shall first describe the algorithm; then, we will analyze its running time; finally, we will show how this algorithm can be recast as a fast parallel algorithm.

The Algorithm

As in Section 3.2, it will suffice, for each prime q such that $q \mid n$, $q \neq p$, to find an irreducible polynomial $f \in \mathbf{F}_p[X]$ of degree m, where m is the order of p mod q, and a q-th nonresidue a in the field $\mathbf{F}_p(\alpha)$, where α is a root of f.

To obtain f, we proceed as follows. If $m > 1$, we recursively invoke the algorithm to construct an irreducible polynomial f of degree m; otherwise, there is nothing to do (we can let f be any linear polynomial).

Let α be a root of f. To obtain a q-th nonresidue in $\mathbf{F}_p(\alpha)$, we proceed as follows. Any element in the field $\mathbf{F}_p(\alpha)$ can be expressed as $g(\alpha)$ where $g \in \mathbf{F}_p[X]$ is a polynomial of degree $< m$. We want to find $g \in \mathbf{F}_p[X]$ such that $g(\alpha)$ is a q-th nonresidue. We do this by examining polynomials $g \in \mathbf{F}_p[X]$ in lexicographic order until $g(\alpha)^{(p^m-1)/q} \neq 1$, where $\sum_{i=0}^{m-1} a_i X^i$ is said to lexicographically precede $\sum_{i=0}^{m-1} b_i X^i$ if for some j, where $0 \leq j \leq m-1$, we have $a_j < b_j$ and $a_i = b_i$ for $i = j+1, \ldots, m-1$.

Complexity Analysis

The running time of this algorithm is determined by the number of recursive calls and the number of polynomials in $\mathbf{F}_p[X]$ that need to be examined in the search for a q-th nonresidue. We will establish bounds for both of these quantities with the following two claims.

Claim 1. No more than $\log n + 1$ *recursive calls are made.* The call tree is reminiscent of the Pratt primality tree [33]. To prove the claim, note that it is the job of each recursive call, other than that associated with n, to construct the q-th cyclotomic extension field for an odd prime q. To do this, it will in the worst case have to recursively construct r-th cyclotomic extensions for each odd prime $r \mid q-1$. We show by induction on q that the total number of calls required to construct the q-th cyclotomic extension is no more than $\log q$. For $q = 3$, exactly one call is made, as no further recursion is necessary. For $q > 3$, the total number is no more than 1 (for the call associated with q itself) plus $\sum_r \log r$ where the sum is taken over all odd primes $r \mid q-1$ (this follows from the induction hypothesis); it is easy to see that this quantity is no more than $\log q$. Thus, to construct an irreducible polynomial of degree n for arbitrary n, the total number of calls made is no more than $1 + \sum_q \log q$, where the sum is taken over all odd primes $q \mid n$; obviously, this quantity is no more than $\log n + 1$. This proves the claim.

Claim 2. Let $d = \lceil 2 \log_p m \rceil$*. If* $0 < d < m$*, then there exists a monic polynomial* $g \in \mathbf{F}_p[X]$ *of degree* d *such that* $g(\alpha)$ *is a* q*-th nonresidue in* $\mathbf{F}_p(\alpha)$*.* This claim is roughly analogous to Ankeny's theorem on the least quadratic nonresidue modulo a prime [3], except that our claim is unconditional, whereas Ankeny's theorem relies on the Extended Riemann Hypothesis. Our claim is a consequence of a theorem of Katz [23, Theorem 2]. Let $\mathbf{F}_{p^d}$ be a finite field with p^d elements, and consider the m-dimensional $\mathbf{F}_{p^d}$-algebra $\mathbf{F}_{p^d}[X]/(f)$. Note that $\mathbf{F}_{p^d}[X]/(f)$ is not a field, unless $\gcd(d, m) = 1$. Note also that $\mathbf{F}_{p^d}[X]/(f)$ is also an algebra over $\mathbf{F}_p[X]/(f)$, since the latter can be embedded in a natural way into the former. The following picture should help to clarify the situation:

$$
\begin{tikzcd}[arrows=dash]
 & \mathbf{F}_{p^d}[X]/(f) \arrow[dl, "m"'] \arrow[dr, "d"] & \\
\mathbf{F}_{p^d} \arrow[dr, "d"'] & & \mathbf{F}_p[X]/(f) \arrow[dl, "m"] \\
 & \mathbf{F}_p &
\end{tikzcd}
$$

We will make use of characters on the group of units of $\mathbf{F}_{p^d}[X]/(f)$. A character on this group is a homomorphism from this group into the nonzero complex numbers. We extend the domain of a character to the entire ring by defining the value of the character to be zero for all non-units.

Suppose that ψ is a nontrivial multiplicative character on $\mathbf{F}_{p^d}[X]/(f)$. Then Katz' theorem implies that

$$\left|\sum_{t \in F_{p^d}} \psi(X - t \bmod f)\right| \leq (m-1)p^{d/2}. \qquad (*)$$

To apply Katz' theorem, we first construct a nontrivial character ψ on $\mathbf{F}_{p^d}[X]/(f)$ in the following way. Let χ be a character of order q on $\mathbf{F}_p(\alpha) = \mathbf{F}_p[X]/(f)$. Let σ be a generator of the Galois group of $\mathbf{F}_{p^d}$ over $\mathbf{F}_p$. By coefficientwise application, σ naturally extends to $\mathbf{F}_{p^d}[X]$. By Galois theory, for any $g \in \mathbf{F}_{p^d}[X]$, $gg^\sigma \cdots g^{\sigma^{d-1}}$ is in $\mathbf{F}_p[X]$. Let $N : \mathbf{F}_{p^d}[X] \to \mathbf{F}_p[X]/(f)$ be defined by $N(g) = gg^\sigma \cdots g^{\sigma^{d-1}} \bmod f$. We want to define the map $\bar{N} : \mathbf{F}_{p^d}[X]/(f) \to \mathbf{F}_p[X]/(f)$ by $\bar{N}(g \bmod f) = N(g)$, but to do this, we must show that this association is well defined. However, one can easily verify that if $g_1 = g + hf$ for some $h \in F_{p^d}[X]$, then $N(g_1) = (g + hf)(g^\sigma + h^\sigma f) \cdots (g^{\sigma^{d-1}} + h^{\sigma^{d-1}} f) \bmod f = N(g)$. Clearly, $\bar{N}$ is a multiplicative map and so we can define the character $\psi = \chi \circ \bar{N}$ on $\mathbf{F}_{p^d}[X]/(f)$. Finally, we want to show that ψ is nontrivial. This is easy to do: choose any $g \in \mathbf{F}_p[X]$ such that $\chi(g \bmod f) \neq 1$; then $\psi(g \bmod f) = \chi(g^d \bmod f) = \chi(g \bmod f)^d$, which is different from 1 since $d \not\equiv 0 \pmod{q}$.

Now consider the left hand side of the inequality $(*)$. Each term of the sum is nonzero since it can be written as $\chi(g \bmod f)$ where g is a nonzero polynomial in $\mathbf{F}_p[X]$ of degree less than m, and hence relatively prime to f.

But, since $d = \lceil 2\log_p m \rceil$, it follows that $d > 2\log_p(m-1)$ which implies that $p^d > (m-1)p^{d/2}$, and hence not all of the terms in the sum can be equal to 1. Therefore, for some $t \in \mathbf{F}_{p^d}$, we have $\psi(X - t \bmod f) \neq 1$. This implies that $\chi((X-t)(X-t^\sigma)\cdots(X-t^{\sigma^{d-1}}) \bmod f)$ is different from 1. But $(X-t)(X-t^\sigma)\cdots(X-t^{\sigma^{d-1}})$ is a monic polynomial in $\mathbf{F}_p[X]$ of degree d, and so the claim is proved.

The second claim implies that if $0 < d < m$, then the number of polynomials we will examine is at most $2p^{\lceil 2\log_p m\rceil} < 2p^{2\log_p m+1} = 2(m^2p)$. Certainly, this bound is valid when $d = 0$ or $d \geq m$ as well. Testing if a candidate polynomial is a q-th power nonresidue can be done with $O((\log p)m^{2+\epsilon})$ $\mathbf{F}_p$-operations. Thus, the search for a q-th nonresidue can be done with $O(p(\log p)m^{4+\epsilon})$ $\mathbf{F}_p$-operations. It then follows from the first claim and the complexity analysis in Section 3.2 that the algorithm in this section for constructing an irreducible polynomial of degree n over $\mathbf{F}_p$ can be performed with $O(p(\log p)n^{4+\epsilon})$ $\mathbf{F}_p$-operations.

A Fast Parallel Algorithm

We use uniform Boolean circuits as our model of parallel computation. We measure the complexity of these circuits (size and depth) as a function of a single input size parameter l. A problem is in NC if there is a uniform family of Boolean circuits of size $l^{O(1)}$ and depth $(\log l)^{O(1)}$ that solves it. For more information on parallel complexity, see the survey article of Karp and Ramachandran [22].

If the characteristic p is small (i.e., $p = l^{O(1)}$), NC algorithms are known for all of the basic operations on polynomials over finite fields we require: addition, subtraction, multiplication, quotient/remainder, and modular exponentiation (see [43] and [20]). In the algorithm in this section, the size of the recursive call tree—and therefore its depth—is $O(\log n)$, and we can search for q-th nonresidues among the $O(m^2p)$ candidates in parallel. Thus, we immediately obtain the following theorem.

Theorem 3.8. *The problem of constructing an irreducible polynomial over $\mathbf{F}_p$ of degree n, given $p, n = l^{O(1)}$, is in NC.*

Remark 1. Using the algorithm of Section 3.4, a similar theorem can be proved for constructing irreducible polynomials over finite extensions of $\mathbf{F}_p$.

Remark 2. The algorithm of Section 3.3 is not easily recast as an NC algorithm. The problem is this: we are required to take $k-1$ successive q-th roots, where k is the highest power to which q divides $p^m - 1$. Typically, k will be quite small, but the author knows of no general bound on k other than the trivial one, $O((\log p)m)$. To obtain an NC algorithm, a bound of $(\log p + \log m)^{O(1)}$ would be required.

Remark 3. The probabilistic algorithms of Rabin [34] and Ben-Or [6] give rise to randomized NC algorithms for constructing irreducible polynomials (provided p is small).

3.6 Open Questions

For fixed p, the probabilistic algorithm of Ben-Or for constructing an irreducible polynomial of degree n over $\mathbf{F}_p$ runs in time $O(n^{2+\epsilon})$, whereas our algorithm requires time $O(n^{4+\epsilon})$ in the worst case. It is natural to ask if this time bound can be improved.

Now consider the construction of Section 3.5. In that section, we searched for a q-th power nonresidue in $\mathbf{F}_p(\alpha) = \mathbf{F}_{p^m}$ by examining polynomials in α over $\mathbf{F}_p$ in lexicographic order. If p is large, in particular $p \geq m^2$, the theory in that section guarantees that one of $\alpha, \alpha+1, \ldots, \alpha+(p-1)$ is a q-th nonresidue in $\mathbf{F}_p(\alpha)$. We ask if, in this situation, we can get a nontrivial bound on the least integer δ such that $\alpha + \delta$ is a q-th power nonresidue, either unconditionally or under the assumption of the ERH.

Finally, the results in this section only address the problem of constructing irreducible polynomials. A more challenging problem is that of finding a primitive polynomial. An irreducible polynomial f is called primitive if α generates the group $\mathbf{F}_p(\alpha)^*$, where α is a root of f. For this problem there are no known polynomial time algorithms, either deterministic or probabilistic, even when p is small.

Bibliography

[1] L. M. Adleman and H. W. Lenstra Jr. Finding irreducible polynomials over finite fields. In *18th Annual ACM Symposium on Theory of Computing*, pages 350–355, 1986.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, 1974.

[3] N. C. Ankeny. The least quadratic nonresidue. *Ann. of Math.*, 55:65–72, 1952.

[4] E. Bach. Realistic analysis of some randomized algorithms. In *19th Annual ACM Symposium on Theory of Computing*, pages 453–461, 1987. Final version to appear, *J. Comput. Sys. Sci.*

[5] E. Bach and V. Shoup. Factoring polynomials with fewer random bits. Technical Report 757, Computer Sciences Department, University of Wisconsin–Madison, 1988. To appear in *J. Symb. Comput.*

[6] M. Ben-Or. Probabilistic algorithms in finite fields. In *22nd Annual Symposium on Foundations of Computer Science*, pages 394–398, 1981.

[7] E. R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, New York, 1968.

[8] E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24(111):713–735, 1970.

[9] A. Borodin and I. Munro. *The Computational Complexity of Algebraic and Numeric Problems*. American Elsevier, 1975.

[10] D. A. Burgess. On character sums and primitive roots. *Proc. London Math. Soc.*, 3(12):179–192, 1962.

[11] D. A. Burgess. On Dirichlet characters of polynomials. *Proc. London Math. Soc.*, 3(13):537–548, 1963.

[12] P. Camion. A deterministic algorithm for factorizing polynomials of $F_q[X]$. *Ann. Discrete Math.*, 17:149–157, 1983.

[13] P. Camion. Improving an algorithm for factoring polynomials over a finite field and constructing large irreducible polynomials. *IEEE Trans. Inform. Theory*, IT-29(3):378– 385, 1983.

[14] D. G. Cantor and E. Kaltofen. Fast multiplication of polynomials over arbitrary rings. Technical Report 87-35, Department of Computer Science, Rensselaer Polytechnic Institute, 1987.

[15] D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36(154):587–592, 1981.

[16] B. Chor and R. L. Rivest. A knapsack type public key cryptosystem based on arithmetic in finite fields. In *Advances in Cryptology: Proceedings of Crypto 84 (Lecture Notes in Computer Science No. 196)*, pages 54–65. Springer-Verlag, 1984.

[17] D. Coppersmith and S. Winograd. Matrix multiplication via Behrend's method. In *19th Annual ACM Symposium on Theory of Computing*, pages 1–6, 1987.

[18] W. Eberly. Very fast parallel matrix and polynomial arithmetic. In *25th Annual Symposium on Foundations of Computer Science*, pages 21–30, 1984.

[19] S. A. Evdokimov. Efficient factorization of polynomials and generalized Riemann hypothesis. Preprint, 1986.

[20] F. Fich and M. Tompa. The parallel complexity of exponentiating polynomials over finite fields. *J. ACM*, 35:651–667, 1988.

[21] M.-D. A. Huang. Riemann hypothesis and finding roots over finite fields. In *17th Annual ACM Symposium on Theory of Computing*, pages 121–130, 1985.

[22] R. M. Karp and V. Ramachandran. A survey of parallel algorithms for shared memory machines. Technical Report 88/408, Comp. Sci. Division, Univ. Calif., Berkeley, 1988. To appear in *Handbook of Theoretical Computer Science*, North-Holland.

[23] N. M. Katz. An estimate for character sums. To appear, *J. AMS*, 1989.

[24] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, second edition, 1981.

[25] J. Lagarias, H. Montgomery, and A. Odlyzko. A bound for the least prime ideal in the Chebotarev density theorem. *Inv. Math.*, 54:271–296, 1979.

[26] S. Lang. *Algebra*. Addison-Wesley, Reading, second edition, 1984.

[27] D. Lazard. On polynomial factorization. In J. Calmet, editor, *Computer Algebra (Lecture Notes in Computer Science No. 144)*, pages 126–134. Springer-Verlag, 1982.

[28] A. K. Lenstra. Factorization of polynomials. In H. W. Lenstra and R. Tijdeman, editors, *Computational Methods in Number Theory*, pages 169–198. Mathematisch Centrum, Amsterdam, 1982.

[29] H. W. Lenstra, 1988. personal communication.

[30] R. Lidl and H. Niederreiter. *Finite Fields*. Addison-Wesley, Reading, 1983.

[31] I. Niven. Formal power series. *Amer. Math. Monthly*, 76:871–889, 1969.

[32] C. Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete Algorithms and Complexity; Proceedings of the Japan-US Joint Seminar*, pages 119–143. Academic Press, 1986.

[33] V. R. Pratt. Every prime has a succinct certificate. *SIAM J. Comput.*, 4:214–220, 1975.

[34] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity, Recent Results and New Directions*, pages 21–39. Academic Press, New York, 1976.

[35] M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9(2):273–280, 1980.

[36] W. Schmidt. *Equations over Finite Fields (Lecture Notes in Mathematics No. 536).* Springer-Verlag, 1976.

[37] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Inform.*, 7:395–398, 1977.

[38] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science*, pages 283–290, 1988. Final version to appear in *Math. Comp.*

[39] V. Shoup. On the deterministic complexity of factoring polynomials over finite fields. Technical Report 782, Computer Sciences Department, University of Wisconsin–Madison, 1988. To appear in *Inform. Process. Lett.*

[40] J. Uspensky. *Theory of Equations.* McGraw-Hill, 1948.

[41] B. L. van der Waerden. *Algebra*, volume 1. Ungar, New York, 1970.

[42] R. R. Varshamov. A general method of synthesizing irreducible polynomials over Galois fields. *Soviet Math. Dokl.*, 29(2):334–336, 1984.

[43] J. von zur Gathen. Parallel algorithms for algebraic problems. *SIAM J. Comput.*, 13(4):802–824, 1984.

[44] J. von zur Gathen. Irreducible polynomials over finite fields. Technical report, Department of Computer Science, University of Toronto, 1986.

[45] J. von zur Gathen. Factoring polynomials and primitive elements for special primes. *Theoret. Comput. Sci.*, 52:77–89, 1987.

[46] J. von zur Gathen and E. Kaltofen. Factorization of multivariate polynomials over finite fields. *Math. Comp.*, 45(171):251–261, 1985.