Fast Construction of Irreducible Polynomials over Finite Fields*

Victor Shoup
Universität des Saarlandes, Saarbrücken, Germany

Abstract

The main result of this paper is a new algorithm for constructing an irreducible polynomial of specified degree n over a finite field $\mathbf{F}_q$. The algorithm is probabilistic, and is asymptotically faster than previously known algorithms for this problem. It uses an expected number of $O\tilde{}(n^2 + n \log q)$ operations in $\mathbf{F}_q$, where the "soft-O" $O\tilde{}$ indicates an implicit factor of $(\log n)^{O(1)}$. In addition, two new polynomial irreducibility tests are described.

1 Introduction

1.1 Statement of main result

Let $\mathbf{F}_q$ be a finite field with q elements, where q is a prime-power. A theorem due to Moore (1893) states that for every positive integer n, there exists a field extension $\mathbf{F}_{q^n}$, unique up to isomorphism, with q^n elements. Such extensions play an important role in coding theory (implementing error correcting codes), cryptography (implementing cryptosystems), and complexity theory (amplifying randomness).

In this paper, we consider the algorithmic version of Moore's theorem: how to efficiently construct this field extension. To implement arithmetic in $\mathbf{F}_{q^n}$, it suffices to have an irreducible polynomial f in $\mathbf{F}_q[X]$ of degree n, since $\mathbf{F}_q[X]/(f)$—the ring of polynomials with arithmetic performed modulo f—is a finite field with q^n elements.

We present a new algorithm for constructing an irreducible polynomial of specified degree n over the finite field $\mathbf{F}_q$. Our algorithm is probabilistic, and is asymptotically faster than previously known probabilistic algorithms for this problem. It uses an expected number of $O\tilde{}(n^2 + n \log q)$ operations in $\mathbf{F}_q$ (additions, subtractions, multiplications, divisions, and zero-tests), where the "soft-O" $O\tilde{}$ indicates an implicit factor of $(\log n)^{O(1)}$. More precisely, our algorithm uses an expected number of

$$O((n^2 \log n + n \log q) \cdot L(n))$$

operations in $\mathbf{F}_q$, where for brevity we use the notation $L(n) = \log n \ \mathrm{loglog}\, n$.

1.2 Discussion of earlier work

One method for constructing an irreducible polynomial of degree n over $\mathbf{F}_q$ is trial and error: choose monic polynomials of degree n at random until an irreducible one is found. This is a reasonable

*Appeared in *Journal of Symbolic Computation* 17, pp. 371–391, 1994; extended abstract in *Proc. 4th Annual Symposium on Discrete Algorithms* (**SODA**), pp. 484–492, 1993.

strategy, since the probability that a randomly chosen monic polynomial of degree n is irreducible is approximately $1/n$. Indeed, just this strategy is recommended in Galois' (1830) paper originating the theory of finite fields.

To turn this idea into an algorithm, one needs to specify an irreducibility test. One such test is due to Butler (1954), which can be implemented so as to use $O\tilde{}(n^{\omega} + n \log q)$ operations in $\mathbf{F}_q$.

Here, ω is such that two $n \times n$ matrices can be multiplied using $O(n^{\omega})$ arithmetic operations. We assume that $2 < \omega \leq 3$. For the "classical" matrix multiplication algorithm, we can take $\omega = 3$; for the asymptotically fast algorithm of Coppersmith & Winograd (1990), we can take $\omega = 2.376$.

Butler's test leads to a probabilistic algorithm for constructing an irreducible polynomial that uses an expected number of $O\tilde{}(n^{\omega+1} + n^2 \log q)$ operations in $\mathbf{F}_q$.

Rabin (1980) suggested using a test for irreducibility that uses $O\tilde{}(n^2 \log q)$ operations in $\mathbf{F}_q$. This yields a probabilistic algorithm for constructing an irreducible polynomial that uses an expected number of $O\tilde{}(n^3 \log q)$ operations in $\mathbf{F}_q$.

Later, Ben-Or (1981) suggested using a different test; while this test has a worst-case running time of $O\tilde{}(n^2 \log q)$ operations, it uses on average only $O\tilde{}(n \log q)$ operations for randomly chosen input polynomials. This yields a probabilistic algorithm for constructing an irreducible polynomial that uses an expected number of $O\tilde{}(n^2 \log q)$ operations.

Recently, a new algorithm for testing irreducibility was presented in von zur Gathen & Shoup (1992). That algorithm uses $O\tilde{}(n^{(\omega+1)/2} + n \log q)$ operations in $\mathbf{F}_q$, yielding a probabilistic algorithm to construct an irreducible polynomial that uses $O\tilde{}(n^{(\omega+3)/2} + n^2 \log q)$ operations. Thus, this new test does not yield a faster method of constructing an irreducible polynomial.

Prior to the results of this paper, Ben-Or's algorithm was asymptotically the fastest. The contrast between the running time of our new algorithm and Ben-Or's algorithm is most striking when q is an n-bit number. In this situation, Ben-Or's algorithm uses $O(n^3 L(n) \log n)$ operations, whereas ours uses $O(n^2 L(n) \log n)$—an order of magnitude improvement.

Our algorithm is not based on the trial and error strategy; nevertheless, its output is still a random monic irreducible polynomial of degree n, uniformly distributed among all such polynomials.

Another feature of our algorithm is its efficient use of randomness. In some situations (e.g., as in Alon *et al.* 1990), it is desirable to limit the amount of randomness used to construct an irreducible polynomial. The only randomness used by our algorithm is an expected number of $O(n)$ randomly chosen elements of $\mathbf{F}_q$, which is optimal to within a constant factor. All of the above algorithms based on trial and error use an expected number of $O(n^2)$ random elements of $\mathbf{F}_q$.

If randomized algorithms are not allowed, then there is no known polynomial-time algorithm for this problem, unless the Extended Riemann Hypothesis is true (Adleman & Lenstra 1986, Evdokimov 1989). There are, however, deterministic polynomial-time algorithms for this problem when q, or even the characteristic of $\mathbf{F}_q$, is small (Chistov 1984, Semaev 1988, Shoup 1990). In particular, in Shoup (1990), a deterministic polynomial-time reduction from this problem to the problem of factoring polynomials in $\mathbf{F}_q[X]$ is given, which leads to a deterministic algorithm for constructing irreducible polynomials that runs in time $\sqrt{p} \cdot (n \log q)^{O(1)}$, where p is the characteristic of $\mathbf{F}_q$.

1.3 Overview

Our algorithm follows in broad outline the deterministic construction in Shoup (1990). A straightforward probabilistic implementation of that algorithm is much slower than either Rabin's or Ben-

Or's. The main technical problems encountered are the computations of traces and minimal polynomials in very large field extensions over $\mathbf{F}_q$. To deal with these problems, we exploit the special structure of these field extensions; we also make use of some of techniques developed in von zur Gathen & Shoup (1992).

The rest of this paper is organized as follows.

In section 2, we briefly discuss some notation and basic facts we will need later.

Sections 3 and 4 contain some new algorithms that will be needed later, but that also may be of independent interest.

In section 3, we present some algorithms for computing minimal polynomials in field extensions and related problems. For example, we present an algorithm that will compute the minimal polynomial over $\mathbf{F}_q$ of an element in $\mathbf{F}_{q^n}$ using $O(n^{(\omega+1)/2})$ operations in $\mathbf{F}_q$.

In section 4, we present an algorithm for factoring a cyclotomic polynomial over $\mathbf{F}_q$. If Φ_n is the nth cyclotomic polynomial, where n is prime, then our algorithm finds an irreducible factor of Φ_n in $\mathbf{F}_q[X]$ using $O\tilde{}(n \log q)$ operations in $\mathbf{F}_q$. The algorithm is a variant, adapted to this special case, of one presented in von zur Gathen & Shoup (1992) that factors a general polynomial using $O\tilde{}(n^2 + n \log q)$ operations in $\mathbf{F}_q$.

In section 5, we prove our main result (Theorem 10).

In section 6, we describe two new algorithms for testing if a given polynomial is irreducible. These algorithms are faster than the algorithm in von zur Gathen & Shoup (1992), but only by a factor of $(\log n)^{O(1)}$. One of the new algorithms is deterministic and the other is probabilistic, erring with probability at most $1/q$.

2 Preliminaries

In this section, we introduce some notation and terminology, and recall some facts that will be needed later.

Let K be a field. Two polynomials of degree at most n in $K[X]$ can be multiplied using $O(nL(n))$ multiplications and additions in K (see Cantor & Kaltofen 1991). Division with remainder involving polynomials of degree at most n can also be done with $O(nL(n))$ operations in K and GCD's can be computed with $O(nL(n)\log n)$ operations (see Aho *et al.* 1974).

We will need to perform arithmetic in certain types of extension rings.

Let f be a monic polynomial in $K[X]$ of degree n, and consider the residue class ring $R = K[X]/(f)$. If θ is the image of X in R, then we have $R = K[\theta]$. Elements in R can be represented as polynomials of degree less than n. With this representation, additions and subtractions in R can be performed with $O(n)$ operations in K, and multiplications with $O(nL(n))$ operations in K. If R is a field, divisions can be performed with $O(nL(n)\log n)$ operations in K. For a polynomial $h \in K[X]$, $(h \bmod f)$ denotes its image in R, while $(h \operatorname{rem} f)$ denotes the remainder of h on division by f.

In this situation, we say that $K[\theta]$ *is the residue class ring defined by* f.

Now consider the cost of performing arithmetic on polynomials in a single variable Y with coefficients in $K[\theta]$. Two polynomials in $K[\theta][Y]$ of degree at most m can be multiplied using $O(nmL(nm))$ operations in K. Moreover, division with remainder involving two such polynomials can be carried out using $O(nmL(nm))$ operations in K, and GCD's can be computed using $O(nmL(nm)\log(nm))$ operations in K.

With these observations, we can calculate the cost of performing arithmetic in a tower of ring extensions. Let g be a monic polynomial of degree m over $K[\theta]$. Consider the residue class ring $S = K[\theta][\Lambda]$ defined by g. Additions and subtractions in S can be performed with $O(N)$ operations in K, where $N = nm$, and multiplications with $O(NL(N))$ operations in K. If S is field, then divisions can be performed with $O(NL(N)\log N)$ operations in K.

Finally, a comment on algorithms and their running times. Typically, we shall present an algorithm that works over an abstract field K or an arbitrary finite field $\mathbf{F}_q$, and state running times in terms of arithmetic operations in the field. Our algorithms also perform Boolean operations, e.g., integer arithmetic, but in all cases the number of bit operations required is dominated by the number of arithmetic operations. We also assume that an algorithm that works over an arbitrary finite field $\mathbf{F}_q$, where q is a prime power p^ℓ, is given the integers p and ℓ as implicit inputs.

3 Algorithms for computing minimal polynomials and related problems

In this section, we present algorithms for computing minimal polynomials and other related problems. These algorithms will be used as subroutines in subsequent sections.

Let K be a field and $K[\theta]$ be a residue class field over K. We will use the following fact concerning the evaluation of a polynomial over K at a point in $K[\theta]$.

Fact 1 *Let K be a field. There is an algorithm with the following properties. It takes as input a residue class ring $K[\theta]$ defined by a polynomial of degree n over K, an element $\alpha \in K[\theta]$, and a polynomial $g \in K[X]$, where $\deg g \leq m$. The algorithm computes $g(\alpha) \in K[\theta]$ using*

$$O(m^{\omega/2}(n/m^{1/2} + 1) + m^{1/2}nL(n))$$

operations in K. In particular, if $m = O(n)$, the algorithm uses $O(n^{(\omega+1)/2})$ operations in K.

Proof. This is essentially Algorithm 2.1 in Brent & Kung (1978). That algorithm computes $g(h) \text{ rem } X^n$ for polynomials g and h in $K[X]$ of degree at most n, but it is easily adapted to meet the specifications above. $\square$

Remark. We will freely make use of a slightly more general version of Fact 1 in which the residue class ring $K[\theta]$ is replaced by a tower of rings $K[\theta][\Lambda]$, and n is replaced by $[K[\theta][\Lambda] : K]$. The proof of this stronger version is straightforward, and we omit it.

The transposition principle. We need to use a fact that says, roughly, the complexity of computing the matrix-vector product $A \cdot v$ is the same as that of computing $v^T \cdot A$, where v^T is the transpose of v. This so-called *transposition principle* can be stated precisely in terms of arithmetic circuits. For our applications, we can restrict ourselves to arithmetic circuits with only addition and multiplication nodes.

Fact 2 *Let K be a field. Suppose that there exists an arithmetic circuit whose inputs are the indeterminates $X_1, \ldots, X_s$ and $Y_1, \ldots, Y_m$. Suppose that the outputs (which are elements of the* *polynomial ring $K[X_1, \ldots, X_s, Y_1, \ldots, Y_m]$) are the entries of the matrix-vector product*

$$A \begin{pmatrix} Y_1 \\ \vdots \\ Y_m \end{pmatrix},$$

where A is an $n \times m$ matrix whose entries are elements of $K[X_1, \ldots, X_s]$.

Then this circuit can be transformed into a circuit with inputs $X_1, \ldots, X_s$ and $Z_1, \ldots, Z_n$, and whose outputs are the entries of

$$(Z_1, \cdots, Z_n)A.$$

The size of the transformed circuit is bounded by a constant times the size of the original circuit. Moreover, there exists an algorithm that will perform the transformation whose running time is also bounded by a constant times the size of the original circuit.

Proof. This was proved in Canny *et al.* (1989) by applying the result of Baur & Strassen (1983). □

In our applications, we want to apply the transposition principle to algorithms—not circuits. In all of the applications in this paper, it is a straightforward matter to run the algorithm that computes $A \cdot v$ to generate a circuit, and then transform the circuit, and finally evaluate the transformed circuit at the desired inputs.

Linearly generated sequences. Our algorithm for computing minimal polynomials in field extensions is based on the theory of *linearly generated sequences.* Let V be a vector space over a field K, and let $\{a_i\}_{i \geq 0}$ be an infinite sequence of elements in V. This sequence is said to be *linearly generated* over K if there exist $c_0, \ldots, c_n \in K$, with $c_n \neq 0$, such that

$$\forall i \geq 0 : c_0 a_i + \cdots c_n a_{i+n} = 0.$$

The polynomial $c_0 + c_1 X + \cdots + c_n X^n \in K[X]$ is called a *generating polynomial* for the sequence. The set of generating polynomials, plus the zero polynomial, form an ideal in $K[X]$, and the monic polynomial that generates it is called the *minimal polynomial* of the sequence.

Fact 3 *Let K be a field, and let $\{a_i\}_{i \geq 0}$ be an infinite sequence of elements in K, linearly generated over K, with minimal polynomial $g \in K[X]$. Then there is an algorithm that takes as input a positive integer m, and the elements $a_0, \ldots, a_{2m-1}$, and computes g, assuming that $m \geq \deg g$. The algorithm can be executed using $O(mL(m) \log m)$ operations in K.*

Proof. The minimal polynomial can be calculated from the quotient sequence of the Euclidean scheme for (f, g), where $f = a_0 X^{2m-1} + \cdots + a_{2m-1} \in K[X]$ and $g = X^{2m} \in K[X]$ (Dornstetter 1987). By using a fast GCD algorithm, all of the computations can be performed using $O(mL(m) \log m)$ operations in K. □

We now present our algorithm for computing minimal polynomials in a field extension $K[\theta]$.

Theorem 4 *Let K be a field. There exists an algorithm that takes as input a residue class field $K[\theta]$ defined by an irreducible polynomial over K of degree n, and an element $\alpha \in K[\theta]$. The algorithm computes the minimal polynomial $g \in K[X]$ of α over K using $O(n^{(\omega+1)/2})$ operations in K. Moreover, if a bound m on the degree of g is given to the algorithm, then it uses only $O(m^{(\omega-1)/2}n + m^{1/2}nL(n))$ operations in K.*

Proof. First, note that the minimal polynomial $g \in K[X]$ of α is the same as the minimal polynomial of the K-linearly generated sequence $\{\alpha^i\}_{i\geq 0}$. Moreover, if $P : K[\theta] \to K$ is a K-linear map, then the sequence $\{P(\alpha^i)\}_{i\geq 0}$ is also linearly generated. Its minimal polynomial divides g, and since g is irreducible, it will equal g so long as this new sequence is not the zero sequence.

Let m be the bound on $\deg g$ if it is given; otherwise, set $m = n$. The algorithm to compute g runs as follows.

1. Let $P : K[\theta] \to K$ be the K-linear map defined by
$$P(1) = 1,\ P(\theta) = \cdots = P(\theta^{n-1}) = 0.$$
Compute the sequence $P(1), P(\alpha), \ldots, P(\alpha^{2m-1})$.

2. Apply the algorithm of Fact 3 to the sequence in step 1 to obtain g, the minimal polynomial of α.

This algorithm is correct, since the sequence $\{P(\alpha^i)\}_{i\geq 0}$ is certainly not the zero sequence, and so the minimal polynomial of this sequence is g.

To execute step 1 of the algorithm, note that we can rephrase this step as that of computing the matrix-vector product
$$(1, 0, \ldots, 0)A,$$
where A is the $n \times 2m$ matrix whose jth column consists of the coordinates of α^{j-1} on the power basis $1, \theta, \ldots, \theta^{n-1}$. By the transposition principle (Fact 2), we can compute this just as fast as an algorithm for computing a matrix-vector product Av, where v is a column vector of dimension $2m$. But this problem is just that of computing $h(\alpha)$, where $h \in K[X]$ is the polynomial of degree at most $2m - 1$ whose coefficients are the entries of v. It follows that we can execute step 1 with $O(m^{(\omega-1)/2}n + m^{1/2}nL(n))$ operations in K.

By Fact 3, step 2 can be executed using $O(mL(m)\log m)$ operations in K.

This proves the theorem. □

Remarks. The idea of using linearly generated sequences over K to compute minimum polynomials for elements in $K[\theta]$ already appears in Rifà & Borrell (1991) and Thiong Ly (1989). Our technical contribution is the improved asymptotic running time that results from applying Brent & Kung's algorithm together with the transposition principle.

We will freely make use of a slightly more general version of Theorem 4 in which the residue class field $K[\theta]$ is replaced by a tower of fields $K[\theta][\Lambda]$, and n is replaced by $[K[\theta][\Lambda] : K]$. The proof of this stronger version is straightforward, and we omit it.

We also note that the condition that the ring extension is actually a field can be dropped by using a *randomized* algorithm whose expected running time is slower than the running time stated in Theorem 4 by a factor of $O(1 + \log n/ \log |K|)$.

We will also need to perform the inverse operation of the algorithm in Fact 1.

Theorem 5 *Let K be a field. There is an algorithm that takes as input a residue class field $K[\theta]$ defined by an irreducible polynomial over K of degree n, and elements $\alpha, \beta \in K[\theta]$. It computes the polynomial $g \in K[X]$ of least degree such that $g(\alpha) = \beta$ if such an polynomial exists; otherwise, it reports failure. The algorithm can be executed using $O(n^{(\omega+1)/2})$ operations in K.*

Proof. The algorithm runs as follows.

1. Compute the minimal polynomial of α over K, and let m be its degree.
2. Let $P : K[\theta] \to K$ be the K-linear map defined by
$$P(1) = 1, \; P(\theta) = \cdots = P(\theta^{n-1}) = 0.$$
Compute the sequence $P(1), P(\alpha), \ldots, P(\alpha^{2m-1})$.
3. Compute the sequence $P(\beta), P(\beta\alpha), \ldots, P(\beta\alpha^{m-1})$.
4. Solve the system of equations
$$\begin{pmatrix} P(1) & \cdots & P(\alpha^{m-1}) \\ \vdots & & \vdots \\ P(\alpha^{m-1}) & \cdots & P(\alpha^{2m-2}) \end{pmatrix} \begin{pmatrix} g_0 \\ \vdots \\ g_{m-1} \end{pmatrix} = \begin{pmatrix} P(\beta) \\ \vdots \\ P(\beta\alpha^{m-1}) \end{pmatrix}.$$
5. Set $g = \sum_{i=0}^{m-1} g_i X^i$, and compute $g(\alpha)$. If $g(\alpha) = \beta$, then g is the desired polynomial; otherwise, there is no such polynomial.

First, we argue for the correctness of this algorithm.

Suppose that $\beta \notin K(\alpha)$. Then the algorithm will certainly report failure in step 5. So assume that $\beta \in K(\alpha)$. Then $m = [K(\alpha) : K]$, where m is as computed in step 1, and there exists a unique polynomial $g = \sum_{i=0}^{m-1} g_i X^i \in K[X]$ such that $g(\alpha) = \beta$. We therefore have
$$\sum_{i=0}^{m-1} g_i \alpha^{i+j} = \beta\alpha^j \quad (0 \le j \le m-1).$$
Applying P to these equations, we get the system of equations in step 4. This is a nonsingular system, since the sequence $\{P(\alpha^i)\}_{i \ge 0}$ is linearly generated over K with minimal polynomial of degree m. Therefore, in step 4, we obtain the coefficients of the unique polynomial $g \in K[X]$ of degree less than m such that $g(\alpha) = \beta$.

Second, we argue for the running time bound.

Step 1 can be executed using $O(n^{(\omega+1)/2})$ operations in K using the algorithm of Theorem 4.

The sequence in step 2 can also be computed using $O(n^{(\omega+1)/2})$ operations in K just as in step 1 in the algorithm in the proof of Theorem 4 (in fact, we have already obtained this sequence as a by-product of executing the algorithm in Theorem 4 in step 1 of this algorithm).

For step 3, notice that by the transposition principle, we can compute this sequence as fast as we can solve an instance of the following problem: given elements α and β in $K[\theta]$, and a polynomial

$h \in K[X]$ of degree less than m, compute $h(\alpha) \cdot \beta$. The latter problem can clearly be solved using $O(n^{(\omega+1)/2})$ operations in K, and so we can also execute step 3 using $O(n^{(\omega+1)/2})$ operations in K.

Step 4 can be executed using the Toeplitz system solver of Brent *et al.* (1980) using $O(mL(m)\log m)$ operations in K.

Step 5 can be executed with the algorithm of Fact 1 using $O(n^{(\omega+1)/2})$ operations in K. $\square$

REMARK. Theorem 5 holds if the residue class field $K[\theta]$ is replaced by a tower of fields $K[\theta][\Lambda]$ and n is replaced by $[K[\theta][\Lambda] : K]$. We will not need this more general version of Theorem 5.

We also need an algorithm to compute norms in field extensions.

Fact 6 *Let K be a field. Then there exists an algorithm that takes as input a residue class field $K[\theta]$ given by an irreducible polynomial over K of degree n, and an element $\alpha \in K[\theta]$. The algorithm computes the norm $N_{K[\theta]/K}(\alpha) \in K$ using $O(nL(n)\log n)$ operations in K.*

Proof. Let $f \in K[X]$ be the monic irreducible polynomial of degree n defining the extension $K[\theta]$. Let $g \in K[X]$ be the polynomial of degree less than n such that $\alpha = g(\theta)$. Suppose that f factors as $f = (X - \theta_1)\cdots(X - \theta_n)$ in the algebraic closure of K. Then we have

$$\begin{aligned} N_{K[\theta]/K}(\alpha) &= g(\theta_1)\cdots g(\theta_n) \\ &= \mathrm{Res}(f, g), \end{aligned}$$

where $\mathrm{Res}(f, g)$ is the resultant of f and g.

As observed in §5 of von zur Gathen (1991), this resultant can be calculated from the quotient sequence of the Euclidean scheme for (f, g). By using a fast GCD algorithm all of the computations can be performed within the stated time bounds. $\square$

REMARK. We will freely make use of a slightly more general version of Fact 6 in which the residue class field $K[\theta]$ is replaced by a tower of fields $K[\theta][\Lambda]$, n is replaced by $[K[\theta][\Lambda] : K]$, and the norm $N_{K[\theta]/K}$ is replaced by $N_{K[\theta][\Lambda]/K[\theta]}$. The proof of this stronger version is straightforward, and we omit it.

4 Factoring cyclotomic polynomials

Let $q = p^\ell$, with p prime, and let r be a prime different from p. In this section, we consider the problem of factoring the rth cyclotomic polynomial $\Phi_r = X^{r-1} + \cdots + X + 1 \in \mathbf{F}_q[X]$. It makes use of the following general fact.

Fact 7 *Let R be a ring of characteristic p. For t a power of p and $k \geq 1$, define the map $T_{t,k} : R \to R$ by*

$$T_{t,k}(\alpha) = \alpha + \alpha^t + \cdots + \alpha^{t^{k-1}}.$$

Then for any $\alpha \in R$, $T_{t,k}(\alpha)$ can be computed using $O(\log k)$ additions in R and $O(\log k)$ powering operations in R of the form $\beta \mapsto \beta^{t^j}$, where $\beta \in R$ and $1 \leq j < k$.

Proof. This fact was observed in von zur Gathen & Shoup (1992). It follows from the identity

$$T_{t,j+k}(\alpha) = T_{t,j}(\alpha) + T_{t,k}(\alpha)^{t^j},$$

utilizing a "repeated doubling" algorithm. □

The algorithm also makes use of the following simple observation.

Fact 8 *Let $g = \sum_{i=0}^{r-1} a_i X^i \in \mathbf{F}_q[X]$ and $k \geq 1$. Then g^{q^k} rem $(X^r - 1)$ can be computed by just permuting the coefficients of g. Namely, if we set $b_i = q^k i$ rem r for $0 \leq i < r$, then*

$$g^{q^k} \text{ rem } (X^r - 1) = \sum_{i=0}^{r-1} a_i X^{b_i}.$$

Theorem 9 *Let $q = p^\ell$, with p prime, and let r be a prime different from p. Then there is a probabilistic algorithm that finds an irreducible factor of $\Phi_r \in \mathbf{F}_q[X]$ using an expected number of $O(rL(r)\log r + rL(r)\log q)$ operations in $\mathbf{F}_q$, and an expected number of $O(r)$ random elements of $\mathbf{F}_q$.*

Proof. The algorithm proceeds as follows.

1. Compute m as the multiplicative order of q modulo r. Set $f = \Phi_r$.
2. If $\deg f = m$, then output f and stop; otherwise, do steps 3–5.
3. Generate a random polynomial $g_1 \in \mathbf{F}_q[X]$ of degree less than $2m$.
4. Compute $g_2 = \sum_{i=0}^{m-1} g_1^{q^i}$ rem f.
5. If q is odd, then do the following:
   (a) Compute $g_3 = g_2^{(q-1)/2}$ rem f.
   (b) Compute $h_1 = \gcd(g_3, f)$, $h_2 = \gcd(g_3 - 1, f)$, and $h_3 = \gcd(g_3 + 1, f)$.
   (c) Replace f by the nonconstant polynomial of least degree among h_1, h_2 and h_3, and go back to step 2.

   Otherwise, do the following:
   (a) Compute $g_3 = \sum_{i=0}^{\ell-1} g_2^{2^i}$.
   (b) Compute $h_1 = \gcd(g_3, f)$ and $h_2 = \gcd(g_3 + 1, f)$.
   (c) Replace f by the nonconstant polynomial of least degree between h_1 and h_2, and go back to step 2.

First of all, note that $m \mid r - 1$ and the irreducible factors of Φ_r are pair-wise nonassociate, each having degree m. Thus we are in the special case of factoring a polynomial whose irreducible factors have the same degree. There are several algorithms in the literature for this problem (Ben-Or 1981, Cantor & Zassenhaus 1981, von zur Gathen & Shoup 1992) and the algorithm presented above is a variant of these. One can easily prove that at each iteration of the main loop (steps 2–5), the polynomial f is replaced by a factor of lower degree with probability at least 1/2. We do not give a proof of this here, but instead refer the reader to one of the papers mentioned above.

Note that whenever the degree of f is reduced, it is reduced by at least a factor of 2. It follows that the expected number of iterations of the main loop is $O(\log r)$.

To attain the running time bounds stated in the theorem, we implement step 4 using the trace map algorithm of Fact 7 with $R = \mathbf{F}_q[X]/(X^r - 1)$, $t = q$, $k = m$, and $\alpha = (g_1 \bmod X^r - 1)$ to compute

$$\hat{g}_2 = \sum_{i=0}^{m-1} g_1^{q^i} \text{ rem } (X^r - 1) \in \mathbf{F}_q[X].$$

To perform the powering operations $\beta \mapsto \beta^{q^j}$ (with $\beta \in R$ and $1 \leq j < m$) required by the trace map algorithm, we use Fact 8. In this way, we can compute $\hat{g}_2$ using $O(r \log r)$ operations in $\mathbf{F}_q$. We then obtain g_2 by one division with remainder that costs $O(rL(r))$ operations in $\mathbf{F}_q$. Since step 4 is executed an expected number of $O(\log r)$ times, the expected number of total operations used to execute step 4 during the course of the algorithm is $O(rL(r) \log r)$.

Suppose that q is odd, and that $s = \deg f$. Then step 5(a) can be executed using $O(sL(s) \log q)$ operations in $\mathbf{F}_q$ with a repeated squaring algorithm, and step 5(b) can be executed using $O(sL(s) \log s)$ operations in $\mathbf{F}_q$ with a fast GCD algorithm. Thus step 5 can be executed using $O(sL(s)(\log s + \log q))$ operations in $\mathbf{F}_q$. Since whenever the degree of f is reduced, it is reduced by a factor of at least 2, it follows that the expected number of total operations used to execute step 5 during the course of the algorithm is $O(rL(r)(\log r + \log q))$.

As can easily be checked, the same bound for the cost of step 5 holds when q is even.

This proves the running time bound. The bound on the expected number of random elements in $\mathbf{F}_q$ is easy to check. $\square$

5 Constructing irreducible polynomials

In this section, we present our main result:

Theorem 10 *There is a probabilistic algorithm that, given n, constructs an irreducible polynomial of degree n over $\mathbf{F}_q$ using an expected number of $O(n^2 L(n) \log n + nL(n) \log q)$ operations in $\mathbf{F}_q$. The output polynomial is uniformly distributed over all monic irreducible polynomials of degree n. The only randomness used by the algorithm is an expected number of $O(n)$ randomly chosen elements in $\mathbf{F}_q$.*

Throughout this section, we assume that $q = p^\ell$, where p is prime.

To prove this theorem, we first treat the situation where n is a prime power, say $n = r^e$, where r is prime. We then consider 3 cases: (1) $r \neq p$ and $r \neq 2$; (2) $r \neq p$ and $r = 2$; (3) $r = p$.

Case 1: $r \neq p$ and $r \neq 2$

Theorem 11 *There is a probabilistic algorithm that, given a prime power $n = r^e$ with $r \neq p$ and $r \neq 2$, constructs an irreducible polynomial of degree n over $\mathbf{F}_q$ using an expected number of $O(n^2 L(n) \log n + rL(r) \log q)$ operations in $\mathbf{F}_q$. The only randomness used by the algorithm is an expected number of $O(n)$ randomly chosen elements in $\mathbf{F}_q$.*

Fact 12 *Let K be a field. Let $d \geq 2$ be an integer, and let $\alpha \in K$, $\alpha \neq 0$. Suppose that for all primes t dividing d, we have $\alpha \notin K^t$, and if $4 \mid d$, then $-1 \in K^2$. Then the polynomial $X^d - \alpha \in K[X]$ is irreducible.*

Proof. This is a direct consequence of Theorem 9.1 on p. 331 of Lang (1984). □

The following is a high-level description of the algorithm in Theorem 11.

Algorithm 13

1. Find an irreducible factor h of the rth cyclotomic polynomial $\Phi_r \in \mathbf{F}_q[X]$. Let $m = \deg h$, and let $\mathbf{F}_q[\theta]$ be the residue class field defined by h.

   Remark. *Note that m is the multiplicative order of q modulo r, and so $m \mid r-1$.*

2. Find an rth power nonresidue $\xi \in \mathbf{F}_q[\theta]$.

   Remark. *By Fact 12 the polynomial $X^n - \xi \in \mathbf{F}_q[\theta][X]$ is irreducible.*

   Let $E = \mathbf{F}_q[\theta][\Lambda]$ be the residue class field defined by $X^n - \xi$.

3. Let H be the (unique) field with $E \supset H \supset \mathbf{F}_q$ and $[H : \mathbf{F}_q] = n$.

   Compute the trace $\Gamma = T_{E/H}(\Lambda)$.

4. Compute the minimal polynomial of Γ over $\mathbf{F}_q$.

   Remark. *In the proof of Theorem 2.1 in Shoup (1990), it is shown that Γ has degree n over $\mathbf{F}_q$; therefore, its minimal polynomial is an irreducible polynomial of degree n in $\mathbf{F}_q[X]$.*

□

The following inclusion diagram illustrates the construction.

$$\begin{tikzcd}
& E = \mathbf{F}_q[\theta][\Lambda] \arrow[dl, dash, "n"'] \arrow[dr, dash, "m"] & \\
\mathbf{F}_q[\theta] \arrow[dr, dash, "m"'] & & H \arrow[dl, dash, "n"] \\
& \mathbf{F}_q &
\end{tikzcd}$$

The correctness of Algorithm 13 follows from the remarks contained therein. To prove Theorem 11, it will suffice to show that each step of Algorithm 13 can be executed within the stated resource bounds.

First of all, note that using the algorithm of Theorem 9, we can execute step 1 of Algorithm 13 using an expected number of $O(rL(r)\log r + rL(r)\log q)$ operations in $\mathbf{F}_q$ and an expected number of $O(r)$ random elements in $\mathbf{F}_q$.

To execute steps 2 and 3 efficiently, we need the following.

Lemma 14 *There is an algorithm that takes as input the residue class field $\mathbf{F}_q[\theta]$ defined in step 1 of Algorithm 13, positive integers k and t, and elements $\alpha, \beta \in \mathbf{F}_q[\theta]$. It is assumed that $\beta = \alpha^{\lfloor q/t \rfloor}$. The algorithm computes $\alpha^{\lfloor q^k/t \rfloor}$ using $O((rL(r) + mL(m)\log t) \cdot \log k)$ operations in $\mathbf{F}_q$.*

Proof. For $u \geq 1$, define integers A_u and B_u by

$$q^u = A_u t + B_u, \quad 0 \leq B_u < t.$$

We are given $\beta = \alpha^{A_1}$ and we want to compute α^{A_k}.

Claim. For any positive integers u and v, we have

$$A_{u+v} = A_v q^u + A_u B_v + \lfloor B_u B_v / t \rfloor.$$

To see why this holds, note that

$$\begin{aligned} q^{u+v} &= q^u(A_v t + B_v) \\ &= q^u A_v t + q^u B_v \\ &= q^u A_v t + (A_u t + B_u) B_v \\ &= (q^u A_v + A_u B_v + \lfloor B_u B_v / t \rfloor) t + (B_u B_v \text{ rem } t). \end{aligned}$$

From the claim, we see that given α^{A_u} and α^{A_v}, we can compute $\alpha^{A_{u+v}}$ by the formula

$$\alpha^{A_{u+v}} = \left(\alpha^{A_v}\right)^{q^u} \cdot \left(\alpha^{A_u}\right)^{B_v} \cdot \alpha^{\lfloor B_u B_v / t \rfloor}.$$

Using Fact 8 and a fast exponentiation algorithm, we can compute $\alpha^{A_{u+v}}$ from α^{A_u} and α^{A_v} with $O(rL(r) + mL(m) \log t)$ operations in $\mathbf{F}_q$.

Therefore, starting with $\beta = \alpha^{A_1}$, and using a "repeated doubling" algorithm, we can compute α^{A_k} with $O((rL(r) + mL(m) \log t) \cdot \log k)$ operations in $\mathbf{F}_q$. $\Box$

Lemma 15 *Step 2 of Algorithm 13 can be executed probabilistically using an expected number of* $O((rL(r) + mL(m) \log r) \cdot \log m + mL(m) \log q)$ *operations in* $\mathbf{F}_q$ *and an expected number of* $O(m)$ *random elements in* $\mathbf{F}_q$.

Proof. To find an rth power nonresidue in $\mathbf{F}_q[\theta]$, we simply choose $\alpha \in \mathbf{F}_q[\theta]$ at random, and test if it is a nonresidue by calculating $\beta = \alpha^{(q^m-1)/r} = \alpha^{\lfloor q^m/r \rfloor}$. If $\beta \neq 1$, α is a nonresidue; otherwise, we choose another α at random and repeat.

To compute β, we first compute $\alpha^{\lfloor q/r \rfloor}$ using a repeated squaring algorithm, which takes $O(mL(m) \log q)$ operations in $\mathbf{F}_q$, and then apply the algorithm of Lemma 14, which takes $O((rL(r) + mL(m) \log r) \cdot \log m)$ operations in $\mathbf{F}_q$.

The expected number of iterations until we find a suitable α is $O(1)$. $\Box$

Lemma 16 *Step 3 of Algorithm 13 can be executed deterministically using*

$$O(nrL(r) \log m + mL(m) \log q)$$

operations in $\mathbf{F}_q$.

Proof. We want to compute

$$\Gamma = T_{E/H}(\Lambda) = \sum_{i=0}^{m-1} \Lambda^{q^{ni}} \in E.$$

We can utilize the trace map algorithm of Fact 7 to compute Γ at a cost of $O(\log m)$ additions in E and $O(\log m)$ powering operations of the form $\Psi \mapsto \Psi^{q^j}$, where $\Psi \in E$ and $1 \leq j < nm$.

We claim that once we precompute $\xi^{\lfloor q/n \rfloor}$, at a cost of $O(mL(m)\log q)$ operations in $\mathbf{F}_q$, we can then perform any one of the above powering operations using $O(nrL(r))$ operations in $\mathbf{F}_q$. The lemma will then follow immediately.

To prove the claim, first recall that any $\Psi \in E$ is represented as

$$\Psi = \sum_{i=0}^{n-1} \alpha_i \Lambda^i,$$

where each α_i is in $\mathbf{F}_q[\theta]$. Then, if $q^j = An + B$, where $0 \leq B < n$, observe that

$$\begin{aligned} \Psi^{q^j} &= \sum_{i=0}^{n-1} (\alpha_i)^{q^j} (\Lambda^{q^j})^i \\ &= \sum_{i=0}^{n-1} (\alpha_i)^{q^j} (\xi^A)^i \Lambda^{Bi}. \end{aligned}$$

The last equation uses the identity $\Lambda^n = \xi$. Thus, we can compute the coordinates of Ψ^{q^j} on the basis $1, \Lambda, \ldots, \Lambda^{n-1}$ by applying the algorithm of Lemma 14 to get ξ^A, and Fact 8 to get $(\alpha_i)^{q^j}$ for each i. Note that replacing Λ^i by Λ^{Bi} for each i just permutes the coordinates. From this, it follows that we can compute Ψ^{q^j} at a cost of $O(nrL(r))$ operations in $\mathbf{F}_q$, the dominant cost being that of computing $(\alpha_i)^{q^j}$ for each i. $\square$

Remark. We observe that Lemma 15 can be generalized as follows (although no use of this generalization is made in this paper). Given an *arbitrary* residue class field $\mathbf{F}_q[\theta]$ of degree m over $\mathbf{F}_q$, and a divisor r of $q^m - 1$, we can find an rth power nonresidue probabilistically using $O\tilde{}(m^{(\omega+1)/2} + m(\log q + \log r))$ operations in $\mathbf{F}_q$. This is done using the polynomial representation of the Frobenius map (as described in von zur Gathen & Shoup (1992)) in conjunction with the technique of Lemma 15.

We now come to step 4 of Algorithm 13. First of all, we remark that we cannot just use the algorithm of Theorem 4, since in the worst case $m = n - 1$, and then that algorithm takes $O(n^{(\omega+3)/2})$ operations in $\mathbf{F}_q$. Instead, we use a "homomorphic imaging" technique: we obtain the values of the minimal polynomial of Γ at roughly n/m points in $\mathbf{F}_q[\theta]$, and then apply the Chinese Remainder Theorem to recover its coefficients. Evaluating the minimal polynomial at a point in $\mathbf{F}_q[\theta]$ turns out to be the same as computing a norm from E down to $\mathbf{F}_q[\theta]$, which can be accomplished using $O\tilde{}(n)$ operations in $\mathbf{F}_q[\theta]$, i.e., $O\tilde{}(nm)$ operations in $\mathbf{F}_q$. Repeating this n/m times (once for each point) takes $O\tilde{}(n^2)$ operations in $\mathbf{F}_q$.

Before describing the details of this step, we state the following estimate.

Fact 17 *Let $k > 1$. The number of elements $\alpha \in \mathbf{F}_{q^k}$ such that $\mathbf{F}_q(\alpha) \neq \mathbf{F}_{q^k}$ is at most $\frac{3}{2}q^{k/2}$. The number of irreducible polynomials over $\mathbf{F}_q$ of degree k is at least $q^k/(2k)$.*

Proof. As is well-known, the number of elements in $\mathbf{F}_{q^k}$ of degree k over $\mathbf{F}_q$ is

$$\sum_{d|k} \mu(d) q^{k/d}.$$

Both statements follow from simple calculations. □

Lemma 18 *Step 4 of Algorithm 13 can be implemented probabilistically with an expected number of $O(n^2 L(n) \log n)$ operations in $\mathbf{F}_q$ and an expected number of $O(n)$ random elements in $\mathbf{F}_q$.*

Proof. The algorithm runs as follows. Set $t = \lceil (n+1)/m \rceil$. Test if $8t^2 m \geq 2^{m/2}$. If so, then calculate the minimal polynomial of Γ using the algorithm of Theorem 4. Since in this case $m = O(\log n)$, this takes $O(n^{(\omega+1)/2} \log n)$ operations in $\mathbf{F}_q$.

So now assume that $8t^2 m < 2^{m/2}$. We then carry out the following steps.

1. Choose elements $\alpha_1, \ldots, \alpha_t \in \mathbf{F}_q[\theta]$ at random.
2. For $1 \leq i \leq t$, compute $g_i \in \mathbf{F}_q[X]$ as the minimal polynomial of α_i over $\mathbf{F}_q$.
3. Compute $G = \prod_{i=1}^t g_i$, and $\gcd(G, G')$, where G' is the formal derivative of G. If this GCD is not 1, or if $\deg G < tm$, go back to step 1; otherwise, proceed to the next step (and note that the g_i's are relatively prime in pairs).
4. For $1 \leq i \leq t$ compute the norm $\beta_i = N_{E/\mathbf{F}_q[\theta]}(\alpha_i - \Gamma)$.
5. For $1 \leq i \leq t$, compute the unique polynomial $h_i \in \mathbf{F}_q[X]$ of degree less than m such that $h_i(\alpha_i) = \beta_i$.
6. Compute the unique polynomial $f \in \mathbf{F}_q[X]$ of degree less than tm that satisfies the system of congruences
$$f \equiv h_i \pmod{g_i} \qquad (1 \leq i \leq t).$$
Then f is the minimal polynomial of Γ over $\mathbf{F}_q$.

We first argue that if the above algorithm terminates, then its output is correct. Let $F \in \mathbf{F}_q[X]$ be the minimal polynomial of Γ. Then we have the well-known formula

$$F = \prod_{j=0}^{n-1} (X - \Gamma^{q^j}).$$

It follows that for any $\alpha \in \mathbf{F}_q[\theta]$,

$$\begin{aligned} F(\alpha) &= \prod_{j=0}^{n-1} (\alpha - \Gamma^{q^j}) \\ &= \prod_{j=0}^{n-1} (\alpha - \Gamma^{q^{mj}}) \qquad (\text{since } \gcd(m, n) = 1) \\ &= \prod_{j=0}^{n-1} (\alpha^{q^{mj}} - \Gamma^{q^{mj}}) \qquad (\text{since } \alpha \in \mathbf{F}_q[\theta]) \\ &= \prod_{j=0}^{n-1} (\alpha - \Gamma)^{q^{mj}} \\ &= N_{E/\mathbf{F}_q[\theta]}(\alpha - \Gamma). \end{aligned}$$

Thus, at step 4, $\beta_i = F(\alpha_i)$. It then follows that at step 5, $h_i \equiv F \pmod{g_i}$. Since $mt \geq n+1$, F is the unique solution to the system of congruences in step 6. We conclude that $F = f$.

To analyze the running time, first consider the loop at steps 1–3. We claim that at each iteration of the loop, the probability that it terminates is at least 1/2. To prove this claim, note that if the loop does not terminate, then at least one of the α_i's lies in a proper subfield of $\mathbf{F}_q[\theta]$, or some pair satisfy the same minimal polynomial. Using the estimates from Fact 17, the probability that any of these events occurs is at most

$$\frac{3t}{2q^{m/2}} + \frac{2t^2m}{q^m} \leq \frac{4t^2m}{2^{m/2}}.$$

But since we are assuming that $8t^2m < 2^{m/2}$, it follows that the probability that any of these events occurs is less than 1/2.

Therefore, the expected number of times the loop at steps 1–3 is executed is no more than 2.

Now consider the cost of executing each step.

1. This step requires only the construction of $O(n)$ random elements in $\mathbf{F}_q$.

2. Using the algorithm of Theorem 4, each individual g_i can be computed using $O(m^{(\omega+1)/2})$ operations in $\mathbf{F}_q$. Repeating this $t = O(n/m)$ times takes $O(nm^{(\omega-1)/2})$ operations in $\mathbf{F}_q$.

3. Using a divide and conquer algorithm, the product polynomial G in this step can be computed with $O(nL(n)\log n)$ operations in $\mathbf{F}_q$. Using a fast GCD algorithm, we can compute $\gcd(G, G')$ in the same time bound.

4. Using the algorithm of Fact 6, each individual norm can be computed using $O(mnL(n)\log n)$ operations in $\mathbf{F}_q$. Repeating this $t = O(n/m)$ times takes $O(n^2L(n)\log n)$ operations in $\mathbf{F}_q$.

5. Using the algorithm of Theorem 5, each individual h_i can be computed using $O(m^{(\omega+1)/2})$ operations in $\mathbf{F}_q$. Repeating this $t = O(n/m)$ times takes $O(nm^{(\omega-1)/2})$ operations in $\mathbf{F}_q$.

6. This step can be executed using a fast Chinese remainder algorithm with $O(nL(n)\log n)$ operations in $\mathbf{F}_q$.

This proves the running time bound. The bound on the number of random elements in $\mathbf{F}_q$ used is clear. $\square$

This completes the proof of Theorem 11.

Cases 2 and 3 can be handled by straightforward adaptations of the algorithms for the corresponding cases in Shoup (1990). Here are the details.

Case 2: $r \neq p$ and $r = 2$

Theorem 19 *There is a probabilistic algorithm that, given $n = 2^e$, with $p \neq 2$, constructs an irreducible polynomial of degree n over $\mathbf{F}_q$ using and expected number of $O(n + \log q)$ operations in $\mathbf{F}_q$. The only randomness used by the algorithm is an expected number of $O(1)$ random elements in $\mathbf{F}_q$.*

Proof. There are two cases.

First, suppose $q \equiv 1 \pmod 4$. Then by Fact 12, the polynomial $X^n - a$ is irreducible provided $a \in \mathbf{F}_q$ is a quadratic nonresidue. Such an a can be find by choosing elements a at random until $a^{(q-1)/2} \neq 1$.

Second, suppose $q \equiv 3 \pmod 4$. In this case, -1 is a quadratic nonresidue in $\mathbf{F}_q$, and so the polynomial $X^2+1 \in \mathbf{F}_q[X]$ is irreducible. If $e = 1$, we are done. Otherwise, let $\mathbf{F}_q[\theta]$ be the residue class field defined by the polynomial X^2+1. By Fact 12, the polynomial $X^{n/2} - \alpha \in \mathbf{F}_q[\theta][X]$ is irreducible provided $\alpha \in \mathbf{F}_q[\theta]$ is not a square. We can find such an α by a randomized search. If β is a root of $X^{n/2} - \alpha$ (in some algebraic closure), then β has degree n over $\mathbf{F}_q$. Writing $\alpha = a + b\theta$, with $a, b \in \mathbf{F}_q$, it is easy to see that the minimal polynomial of β over $\mathbf{F}_q$ is

$$X^n - 2aX + (a^2 + b^2) \in \mathbf{F}_q[X].$$

□

Case 3: $r = p$

Theorem 20 *There is a probabilistic algorithm that, given $n = p^e$, constructs an irreducible polynomial over $\mathbf{F}_q$ of degree n using an expected number of $O(n^{(\omega+1)/2} + \log q)$ operations in $\mathbf{F}_q$. The only randomness used by the algorithm is an expected number of $O(1)$ random elements in $\mathbf{F}_q$.*

Proof. The algorithm runs as follows.

1. Choose $a \in \mathbf{F}_q$ at random.

2. Compute the trace $b = T_{\mathbf{F}_q/\mathbf{F}_p}(a)$.

3. If $b = 0$, go back to step 1; otherwise, set $f = X^p - X - a \in \mathbf{F}_q[X]$ and proceed to the next step.

   Remark. *At this point, the polynomial $f \in \mathbf{F}_q[X]$ is irreducible (this follows from Hilbert's Theorem 90 and the Artin-Schreier Theorem on p. 325 of Lang 1984).*

4. For $j = 2, \ldots, e$, do steps 5–6.

5. Let $\mathbf{F}_q[\theta]$ be the residue class field defined by the polynomial f. If $j = 2$, set $\alpha = \theta^{p-1} \in \mathbf{F}_q[\theta]$; otherwise, set $\alpha = \theta^{2p-1} - \theta^p \in \mathbf{F}_q[\theta]$.

   Remark. *At this point, the polynomial $X^p - X - \alpha \in \mathbf{F}_q[\theta][X]$ is irreducible (this follows from Lemma 5 of Adleman & Lenstra 1986).*

6. Let $\mathbf{F}_q[\theta][\Lambda]$ be the residue class field defined by the polynomial $X^p - X - \alpha \in \mathbf{F}_q[\theta][X]$. Compute the minimal polynomial of Λ over $\mathbf{F}_q$, and replace f with this polynomial.

   Remark. *At this point, f is an irreducible polynomial over $\mathbf{F}_q$ of degree p^j.*

7. Output f.

   Remark. *At this point, f is an irreducible polynomial of degree p^e.*

The correctness of the algorithm is easily checked. We omit the details.

For the running time bound, note that the expected number of iterations of the loop at steps 1–3 is $O(1)$. The trace can be calculated using $O(\log q)$ operations in $\mathbf{F}_q$ by using a repeated squaring algorithm in conjunction with the formula $T_{\mathbf{F}_q/\mathbf{F}_p}(a) = \sum_{i=0}^{e-1} a^{p^i}$.

Next, note that for $2 \leq j \leq e$, the jth iteration of step 6 can be performed using $O(p^{j(\omega+1)/2})$ operations in $\mathbf{F}_q$ by employing the algorithm of Theorem 4. Summing from 2 to e gives $O(n^{(\omega+1)/2})$ operations in $\mathbf{F}_q$. □

Proof of Theorem 10

We now have everything we need to prove Theorem 10.

The algorithm runs as follows.

Algorithm 21 1. Decompose n as $n = n_1 \cdots n_k$, where each n_i is a prime power, and the n_i's are relatively prime in pairs.

2. Construct an irreducible polynomial f of degree n_1.

3. For $i = 2, \ldots, k$, do steps 4–5.

4. Construct an irreducible polynomial g of degree n_i over $\mathbf{F}_q$.

5. Remark. *At this point, f is an irreducible polynomial in $\mathbf{F}_q[X]$ of degree $n_1 \cdots n_{i-1}$ and g is an irreducible polynomial in $\mathbf{F}_q[X]$ of degree n_i.*

   Let $\mathbf{F}_q[\theta]$ be the residue class ring defined by f.

   Remark. *Note that g remains irreducible over the extension field $\mathbf{F}_q[\theta]$ (see, e.g., Theorem 3.46 in Lidl & Niederreiter 1983).*

   Let $E[\theta][\Lambda]$ be the residue class field defined by g.

   Remark. *Note that $\Lambda + \theta$ has degree $n_1 \cdots n_i$ over $\mathbf{F}_q$ (this follows from Lemma 7 of Adleman & Lenstra 1986).*

   Compute the minimal polynomial of $\Lambda + \theta$ over $\mathbf{F}_q$, and replace f with this polynomial.

6. Remark. *At this point, $f \in \mathbf{F}_q[X]$ is an irreducible polynomial of degree n.*

   Let $\mathbf{F}_q[\theta]$ be the residue class field defined by f. Choose $\alpha \in E[\theta]$ at random, and compute its minimal polynomial h.

7. If $\deg h < n$, go back to step 6; otherwise, output h and stop.

□

The correctness of the algorithm is clear.

The running time bounds are easy to obtain from the following observations. To construct the irreducible polynomials of prime power degrees, we can use the algorithms of of Theorems 11, 19, and 20. To construct the minimal polynomials, we can use the algorithm of Theorem 4. Also, note that by Fact 17, the expected number times the loop at steps 6–7 is executed is $O(1)$.

It is also easy to check the bound on the use of randomness.

This completes the proof of Theorem 10

6 Two irreducibility tests

In this section we consider the problem of testing if a given polynomial over $\mathbf{F}_q$ of degree n is irreducible. Let r denote the number of distinct primes dividing n. The irreducibility test in von zur Gathen & Shoup (1992) uses

$$O(n^{(\omega+1)/2} r \log n + nL(n)\log q)$$

operations in $\mathbf{F}_q$. In this section, we present two new tests. The first is deterministic and uses

$$O(n^{(\omega+1)/2} \log r \log n + nL(n)\log q)$$

operations in $\mathbf{F}_q$. The second is probabilistic, uses

$$O(n^{(\omega+1)/2} \log n + nL(n)\log q)$$

operations in $\mathbf{F}_q$, makes no errors on irreducible inputs, and errs with probability at most $1/q$ when the input is reducible.

The first test is based on the following lemma.

Lemma 22 *Let G be a group and $\sigma \in G$. Let $n \geq 1$ with prime factorization*

$$n = p_1^{e_1} \dots p_r^{e_r}.$$

Then we can compute

$$\sigma^{n/p_1}, \ \dots, \sigma^{n/p_r}$$

using $O(\log r \cdot \log n)$ multiplications in G.

Proof. First compute $\tau_i = \sigma^{n/p_i^{e_i}}$ for $1 \leq i \leq r$ recursively as follows. If $r = 1$, there is nothing to do. Otherwise, let

$$s = \lfloor r/2 \rfloor, \quad n_1 = \prod_{i=1}^{s} p_i^{e_i}, \quad n_2 = \prod_{i=s+1}^{r} p_i^{e_i}.$$

First, compute $\sigma_1 = \sigma^{n_1}$ and $\sigma_2 = \sigma^{n_2}$ using a repeated squaring algorithm. Then, recursively compute

$$\sigma_2^{n_1/p_i^{e_i}} \quad (1 \leq i \leq s) \qquad \text{and} \qquad \sigma_1^{n_2/p_i^{e_i}} \quad (s+1 \leq i \leq r).$$

One sees that the recursion depth is $O(\log r)$, and the number of multiplies at each level of the corresponding recursion tree is $O(\log n)$. Thus we can compute τ_i for $1 \leq i \leq r$ with $O(\log r \cdot \log n)$ multiplies in G.

Finally, for each i, we compute σ^{n/p_i} by raising τ_i to the power $p_i^{e_i - 1}$. The cost of this step is $O(\log n)$ multiplies. □

Theorem 23 *There is a deterministic algorithm that takes as input a polynomial f of degree n over $\mathbf{F}_q$ and determines if it is irreducible. The algorithm uses*

$$O(n^{(\omega+1)/2} \log r \log n + nL(n)\log q)$$

operations in $\mathbf{F}_q$, where r is the number of distinct primes dividing n.

Proof. Let $n = p_1^{e_1} \cdots p_r^{e_r}$ be the prime factorization of n. To test for irreducibility, it suffices to test if $X^{q^n} \equiv X \pmod{f}$, and $\gcd(X^{q^{n/p_i}} - X, f) = 1$ for $1 \le i \le r$.

The dominant cost of this computation is that of computing $X^{q^n} \bmod f$, and $X^{q^{n/p_i}} \bmod f$ for $1 \le i \le r$. To compute these quantities, we first compute $X^q \bmod f$ by repeated squaring. Then we apply the above lemma, taking σ to be the q-th power map on $\mathbf{F}_q[X]/(f)$, and G to be the group of $\mathbf{F}_q$-automorphisms on $\mathbf{F}_q[X]/(f)$. Automorphisms are represented by their action on $(X \bmod f)$. Therefore, given representations for two automorphisms, we can compute the representation of their composition using $O(n^{(\omega+1)/2})$ operations in $\mathbf{F}_q$ with the algorithm in Fact 1. $\square$

Let p be the characteristic of $\mathbf{F}_q$, and let $f \in \mathbf{F}_q[X]$ be a polynomial of degree n. For $g \in \mathbf{F}_q[X]$, define

$$T(g) = \sum_{i=0}^{n-1} g^{q^i} \text{ rem } f.$$

Lemma 24 *Notation as above. f is reducible if and only if*

(a) $T(g) \notin \mathbf{F}_q$ for some $g \in \mathbf{F}_q[X]$, or

(b) $p \mid n$ and $X^{q^{n/p}} \equiv X \pmod{f}$.

Proof. If f is irreducible, then T is just the trace map from $\mathbf{F}_{q^n}$ to $\mathbf{F}_q$, so condition (a) cannot hold. Condition (b) also cannot hold; otherwise, all irreducible factors of f have degree dividing n/p, contradicting the irreducibility of f.

Now suppose f is reducible.

First suppose that f is not squarefree. Then $T(X) \notin \mathbf{F}_q$. To see this, suppose the contrary. Then we would have an equation

$$\sum_{i=0}^{n-1} X^{q^i} = c + fh,$$

where $c \in \mathbf{F}_q$ and $h \in \mathbf{F}_q[X]$. The derivative of the left-hand side is 1, whereas the derivative of the right-hand side is not, as fh is not squarefree. Thus, condition (a) of the theorem must hold.

So now assume that f is squarefree and write $f = f_1 \cdots f_k$, where the f_i's are irreducible polynomials. There are two cases.

Case 1. Suppose that $\deg(f_i) \nmid n$ for some i. Let d be the degree of this f_i. Choose a polynomial g that generates a normal basis for $\mathbf{F}_q[X]/(f_i)$ over $\mathbf{F}_q$. Then $T(g)$ cannot be a constant modulo f_i. To see this, suppose the contrary. Write $n = du + v$, $0 < v < d$. We then would have an equation

$$u \sum_{i=0}^{d-1} g^{q^i} + \sum_{i=0}^{v-1} g^{q^i} \equiv c \bmod f_i,$$

where $c \in \mathbf{F}_q$. The first sum on the left-hand side is a constant modulo f_i, and the right-hand side is also a constant. However, since g generates a normal basis, the second sum on the left-hand side cannot be a constant modulo f_i. Thus $T(g)$ is not a constant, and condition (a) holds.

Case 2. Suppose that $\deg(f_i) \mid n$ for all i. Let $d_i = \deg(f_i)$ for each i. Then modulo f_i, T acts as

$$n/d_i \cdot T_{\mathbf{F}_{q^{d_i}}/\mathbf{F}_q}.$$

If $p \mid n/d_i$ for all i, then condition (b) holds. Otherwise, choose some i such that $p \nmid n/d_i$. Choose any $i' \neq i$. Then with the Chinese Remainder Theorem, we can choose $g \in \mathbf{F}_q[X]$ such that $T(g)$ is 1 modulo f_i and is 0 modulo $f_{i'}$. Thus, $T(g)$ is not a constant, and condition (a) holds. □

To make this theorem into a probabilistic test, observe that the polynomials $g \in \mathbf{F}_q[X]$ of degree less than n such that $T(g) \in \mathbf{F}_q$ form a vector space. If f is reducible and condition (b) does not hold, then this vector space is a proper subspace of the space of all polynomials of degree less than n. Therefore, in this case, a random polynomial g of degree less than n will satisfy $T(g) \in \mathbf{F}_q$ with probability at most $1/q$. This suggests the following algorithm.

Algorithm 25 *If $p \mid n$, test if $X^{q^{n/p}} \equiv X \pmod{f}$. If so, output "reducible" and stop. Otherwise, choose a random polynomial $g \in \mathbf{F}_q[X]$ of degree less than n, and test if $T(g) \in \mathbf{F}_q$. If so, output "irreducible" and otherwise output "reducible."*

Theorem 26 *Algorithm 25 can be implemented so as to use*

$$O(n^{(\omega+1)/2} \log n + nL(n) \log q)$$

operations in $\mathbf{F}_q$. If the input f is irreducible, its output is always correct; otherwise, the ouput is correct with probability at least $1 - 1/q$.

Proof. The correctness follows from the above theorem and the remarks following it. The stated running time can be achieved using Algorithm 5.2 in von zur Gathen & Shoup (1992). □

7 Conclusion

We have presented a probabilistic algorithm that constructs an irreducible polynomial of degree n over the finite field $\mathbf{F}_q$ using an expected number of $\tilde{O}(n^2 + n \log q)$ operations in $\mathbf{F}_q$. Our algorithm is asymptotically faster than previously known methods, which use an expected number of $\tilde{O}(n^2 \log q)$ operations.

As an open problem, we ask if one can do this faster, e.g., with an expected number of $\tilde{O}(n^\kappa + n \log q)$ for some $\kappa < 2$.

As another open problem, we ask if we can construct an irreducible polynomial of degree n over $\mathbf{F}_2$ *deterministically* in time $\tilde{O}(n^2)$. The algorithm in Shoup (1990) runs in time $\tilde{O}(n^4)$. Using the techniques in this paper, it is possible to reduce this to $\tilde{O}(n^3)$; however, at the present time it is not clear how to do better.

We have also presented two new tests for irreducibility.

The problem of computing $g(h) \bmod f$, for polynomials f, g, and h, is clearly an important problem, as demonstrated by the many applications in this paper and in von zur gathen & Shoup (1992). It would be advantageous to find an algorithm for this problem that is faster than the $O(n^{(\omega+1)/2})$ algorithm we have used here.

References

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

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

N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *31st Annual Symposium on Foundations of Computer Science*, pp. 544–553, 1990.

W. Baur and V. Strassen. The complexity of computing partial derivatives. *Theoretical Computer Science* **22**, pp. 317–330, 1983.

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

R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. Assoc. Comput. Mach.* **25**, pp. 581–595, 1978.

R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *J. Algorithms* **1**, pp. 259–295, 1980.

M. C. R. Butler. On the reducibility of polynomials over a finite field. *Quart. J. Math., Oxford Ser. (2)* **5**, pp. 102–107, 1954.

J. F. Canny, E. Kaltofen, and L. Yagati. Solving systems of non-linear polynomial equations faster. In *Proc. ACM-SIGSAM Int. Symp. on Symbolic and Algebraic Computation*, pp. 121–128, 1989.

D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary rings. *Acta Inform.* **28**, pp. 693–701, 1991.

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

A. L. Chistov. Polynomial time construction of a finite field. In *Abstracts of Lectures at 7th All-Union Conference in Mathematical Logic, Novosibirsk*, p. 196, 1984. In Russian.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comp.* **9**(3), pp. 23–52, 1990.

J. L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory* **IT-33**, pp. 428–431, 1987.

S. A. Evdokimov. Factoring a solvable polynomial over a finite field and Generalized Riemann Hypothesis. *Zapiski Nauchn. Semin. Leningr. Otdel. Matem. Inst. Acad. Sci. USSR* **176**, pp. 104–117, 1989. In Russian.

E. Galois. Sur la théorie des nombres. In *Écrits et Mémoires Mathématiques d'Évariste Galois*, ed. R. Bourgne and J.-P. Arza, pp. 112–128. Gauthier-Villars, 1830.

J. von zur Gathen. Tests for permutation polynomials. *SIAM J. Comput.* **20**, pp. 591–602, 1991.

J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity* **2**, pp. 187–224, 1992.

S. Lang. *Algebra.* Addison-Wesley, second edition, 1984.

R. Lidl and H. Niederreiter. *Finite Fields.* Addison-Wesley, 1983.

E. H. Moore. A doubly-infinite system of simple groups. *Bull. New York Math. Soc.* **3**, pp. 73–78, 1893.

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

J. Rifà and J. Borrell. Improving the time complexity of the computation of irreducible and primitive polynomials in finite fields. In *Proc. AAECC-9, Lecture Notes in Computer Science 539*, vol. 12, pp. 352–359, 1991.

I. A. Semaev. Construction of irreducible polynomials over finite fields with linearly independent roots. *Mat. Sbornik* **135**, pp. 520–532, 1988. In Russian; English translation in *Math. USSR–Sbornik*, 63(2):507–519, 1989.

V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.* **54**(189), pp. 435–447, 1990.

A. Thiong Ly. Note for computing the minimum polynomial of elements in large finite fields. In *Coding Theory and Applications, Lecture Notes in Computer Science 388*, pp. 185–192, 1989.