Factoring Polynomials Over Finite Fields

Daniel Panario
University of Toronto

January 23, 1995

[summary by Reynald Lercier]

Let $q = p^m$ where p is a prime and $m \in \mathbb{N}^*$, let f be a monic univariate polynomial of degree n in $\mathbb{F}_q[X]$; this talk surveys known algorithms to find the complete factorization $f = f_1^{e_1} \cdots f_r^{e_r}$ where the f_i's are monic distinct irreducible polynomials and where $e_i \in \mathbb{N}^*$ for $i \in \{1, \ldots, r\}$. This problem plays an important role in various fields like computer algebra, cryptography, number theory, coding theory, ...

First, we present the main ideas behind Berlekamp's algorithm (Section 1). Then, we give a general factoring algorithm composed of three stages (squarefree, distinct-degree and equal-degree factorization) which provides a framework for several other algorithms (Section 2). Finally, we outline best known asymptotic complexity, current bottlenecks and recent results (Section 3).

To compare these algorithms, the unit cost will be multiplication in $\mathbb{F}_q$. Moreover, the arithmetic considered is fast for polynomial algorithms and classical for linear algebra.

Notation. Throughout the summary, we let $O^\sim(n) = O(n \log^k(n))$ where k is a constant.

1. Berlekamp algorithm

Berlekamp's ideas [1] lead to an efficient algorithm to factorize a polynomial f with no repeated factors. Let R be the polynomial ring $\mathbb{F}_q[X]/(f)$ and R_i be the polynomial rings $\mathbb{F}_q[X]/(f_i)$ for $i \in \{1, \ldots, r\}$, then

$$R \simeq R_1 \times \cdots \times R_r$$

by the Chinese Remainder Theorem. We concentrate now on the Frobenius map Φ on R given by

$$\begin{aligned} \Phi : R &\longrightarrow R, \\ h &\longmapsto h^q. \end{aligned}$$

The set of fixed points of Φ is

$$B = \{h \in R, h^q = h\}$$

and again by the Chinese Remainder Theorem, we have

$$B \simeq \mathbb{F}_q^r.$$

This means that B is a $\mathbb{F}_q$ vector space of dimension r, the number of irreducible polynomials f_i. Berlekamp proved the following theorem.

Theorem 1. *Let $h \in B$, then*

$$f = \prod_{\alpha \in \mathbb{F}_q} \gcd(f, h - \alpha).$$

In fact, we obtain a non trivial factorization of f for $\deg h \geq 1$ (if $\deg h = 0$, we obtain f multiplied by 1). Definition 1 outlines an important property of B.

DEFINITION 1. A set $S = \{h_1, \ldots, h_s\}$ is called a separating set for f if for any two distinct irreducible factors f_i and f_j, there exists $h_k \in S$ and $\alpha_k \in \mathbb{F}_q$ such that $h_k - \alpha_k$ is divisible by f_i, but not f_j.

Theorem 2 connects Theorem 1 with Definition 1.

THEOREM 2. *Let $\{1, v_2, \ldots, v_r\}$ be a polynomial basis of B, then $\{v_2, \ldots, v_r\}$ is a separating set for f.*

These results finally lead to the following algorithm whose complexity is $O^{\sim}(n^3 + qn^2)$.

(1) Form the matrix of the mapping $\Phi - \mathrm{Id}$ with respect to the basis $\{1, X, \ldots, X^{n-1}\}$ of R;
(2) Obtain a basis $\{1, v_2, \ldots, v_r\}$ of $B = \ker(\Phi - \mathrm{Id})$ using Gaussian elimination;
(3) Factor f computing

$$f = \prod_{\alpha \in \mathbb{F}_q} \gcd(f, v_2 - \alpha)$$

and refine the partial factorization successively using $v_3, \ldots, v_r$ until the complete factorization of f is obtained.

2. A general factoring algorithm

We focus now on a different approach for factoring polynomials over finite fields. It is a general method that breaks the problem into three subproblems.

Square-free factorization: Find monic square-free pairwise relatively prime polynomials $g_1, \ldots, g_n$ such that

$$f = g_1 g_2^2 \cdots g_n^n.$$

Distinct degree factorization: Split a square-free polynomial into polynomials the irreducible factors of which have the same degree.

Equal degree factorization: Completely factor a polynomial the irreducible polynomial factors of which have the same degree.

2.1. Square-free factorization.

This subproblem is solved easily by more or less computing the gcd of f with its derivative f'. More rigourously, a possible algorithm is the following [3].

```
$i := 1$; $R := 1$; $a := f$; $b := f'$; $c := \gcd(a, b)$; $w := a/c$;
while $c \neq 1$ do
    $y := \gcd(w, c)$; $z := w/y$; $R := R * z^i$; $i := i + 1$; $w := y$; $c := c/y$
$R := R * w^i$;
return($R$)
```

This method has cost $O^{\sim}(n)$.

2.2. Distinct degree factorization. This subproblem is solved by Theorem 3 [4].

THEOREM 3. *For every $i \in \mathbb{N}$, the product of all monic irreducible polynomials over $\mathbb{F}_q$ whose degrees divide i is equal to $X^{q^i} - X$.*

This theorem leads to the following algorithm which returns a n-tuple $(g_1, \ldots, g_n)$ where each polynomial g_i contains all the factors of degree i of f.

```
(1) Set $h_0 = X$ and $f_0 = f$;
(2) For $i = 1, \ldots, n$, do
        – Compute $h_i = h_{i-1}^q \bmod f$;
        – Compute $g_i = \gcd(h_i - X, f_{i-1})$ and $f_i = f_{i-1}/g_i$;
(3) Return $(g_1, \ldots, g_n)$.
```

The main cost of this algorithm is the computation of X^{q^i} for $i = 1, \ldots, n$. By repeated squaring, this cost is $O\tilde{}\,(n^2 \log q)$. A better way of computing this quantity consists in using an algorithm to iterate the Frobenius map. At first we compute $X^q \bmod f$ and then go doubling evaluating $X^{q^i} \bmod f$ in X^q to obtain $X^{q^{i+1}}$. So $X^{q^2} \bmod f = X^q(X^q) \bmod f$, $X^{q^3} \bmod f = X^{q^2}(X^q) \bmod f$, and so on. The cost of this method is $O\tilde{}\,(n^2 + n \log q)$

2.3. Equal degree factorization. The probabilistic algorithm we are going to describe to find the r irreducible factors $f_1, \ldots, f_r$ of degree d of a polynomial f of degree $n = dr$ in $\mathbb{F}_q$ with q odd is due to Cantor-Zassenhaus [2].

If there exists a polynomial $c \in \mathbb{F}_q[X]$ such that $c \bmod f_i = 0$ and $c \bmod f_j \neq 0$ for $1 \leq i < j \leq r$, then $\gcd(c, f)$ splits f. To take advantage of this idea, we choose at random a polynomial a of $\mathbb{F}_q[X]/(f)$. Then the polynomials $a_i = a \bmod f_i$ for $1 \leq i \leq r$ are independent and uniformly distributed elements in $\mathbb{F}_q[X]/(f_i)$. So

$$a_i^{\frac{q^d-1}{2}} \equiv \pm 1 \bmod f_i$$

with probability $1/2$ each. Consequently, $a^{\frac{q^d-1}{2}} - 1$ does not factor f with probability $2(1/2)^r$. This idea leads to the following algorithm which returns one factor of f or FAIL.

```
(1) Choose $a \in \mathbb{F}_q[X]/(f)$ at random;
(2) Compute $g = \gcd(a, f)$. If $g \neq 1$, then return $g$;
(3) Compute $b = a^{\frac{q^d-1}{2}} - 1 \bmod f$;
(4) Compute $g = \gcd(b, f)$. If $g \neq 1$, then return $g$ else return FAIL.
```

Its cost is $O\tilde{}\,(n^2 \log q)$.

3. Recent results

New results improve the scheme described in Section 2.

- J. von zur Gathen and V. Shoup [7]: Distinct degree factorization: $O\tilde{}\,(n^2 + n \log q)$; equal degree factorization: $O\tilde{}\,(n^{1.7} + n \log q)$.
- E. Kaltofen and V. Shoup announced in [6]: Distinct degree factorization: $O(n^{1.815} \log q)$ asymptotically but $O(n^{2.5} + n \log q)$ in practice; equal degree factorization: $O(n^2 \log n + n \log q)$ in practice.

Nevertheless, the main cost remains the computation of X^{q^i} for $i = 1, \ldots, n$. Furthermore, other recent results must be cited.

- Evdokimov (1993): A deterministic algorithm, quasi polynomial time $(n^{\log n} \log q)^{O(1)}$.
- Niederreiter [5]: A new deterministic algorithm.

- Kaltofen-Lobo (1994): Randomized Berlekamp with Wiedemann's linear solver, in time $O^{\sim}(n^2 + n \log q)$.

Bibliography

[1] Berlekamp (E. R.). – Factoring polynomials over large finite fields. *Mathematics of Computation*, vol. 24, n° 111, 1970, pp. 713–735.
[2] Cantor (D. G.) and Zassenhaus (H.). – A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, vol. 36, 1981, pp. 587–592.
[3] Geddes (Keith O.), Czapor (Stephen R.), and Labahn (George). – *Algorithms for Computer Algebra*. – Kluwer Academic Publishers, 1992.
[4] Lidl (Rudolf) and Niederreiter (Harald). – *Finite Fields*. – Addison-Wesley, 1983, *Encyclopedia of Mathematics and its Applications*, vol. 20.
[5] Niederreiter (H.). – Factoring polynomials over finite fields using differential equations and normal bases. *Mathematics of Computation*, vol. 62, 1994, pp. 819–830.
[6] Shoup (V.). – A new polynomial factorization algorithm and its implementation. – 1994. Preprint.
[7] von zur Gathen (J.) and Shoup (V.). – Computing Frobenius map and factoring polynomials. *Computational Complexity*, vol. 2, 1992, pp. 187–224.