COMBINATORICA
Akadémiai Kiadó — Springer-Verlag

COMBINATORICA 9 (2) 1989 199—206

FACTORING POLYNOMIALS MODULO SPECIAL PRIMES

L. RÓNYAI*

Received January 15, 1988

We consider the problem of factoring polynomials over $GF(p)$ for those prime numbers p for which all prime factors of $p-1$ are small. We show that if we have a primitive t-th root of unity for every prime t dividing $p-1$ then factoring polynomials over $GF(p)$ can be done in deterministic polynomial time.

1. Introduction

J. von zur Gathen [1987] considered the problem of factoring polynomials over $GF(p)$ in the case when $p-1$ has small prime factors. Following his definition, the *smoothness* $S(k)$ of an integer k is the largest prime factor of k. He showed that the problem of factoring polynomials over $GF(p)$ and the problem of finding a primitive element in $GF(p)$ are polynomial time equivalent via Cook reductions. Here "polynomial time" means polynomial time in the input size plus $S(p-1)$. Also he proved that if one assumes the Extended Riemann Hypothesis (ERH), then primitive elements can be found in time $(\log p+S(p-1))^{O(1)}$. Thus, under ERH, one can factor polynomials over $GF(p)$ using $(n+\log p+S(p-1))^{O(1)}$ bit operations, where n is the degree of the polynomial to be factored.

Before formulating our result, we introduce some notation. In this paper p denotes an odd prime and

$$p-1 = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$$

denotes the prime factorization of $p-1$. The *socle* $\operatorname{soc}(p)$ of $GF(p)$ is defined as

$$\operatorname{soc}(p) = \{\zeta \in GF(p),\ \zeta \text{ is a } t\text{-th root unity for some prime } t \mid p-1\}.$$

Clearly we have $m=|\operatorname{soc}(p)|=p_1+p_2+\dots+p_r-r+1 \leqq S(p-1)\log p$.

In this note we prove the following

Theorem 1.1. *Let p be an odd prime and suppose that* $\operatorname{soc}(p)$ *is given. Then we can factor polynomials over $GF(p)$ in time $(n+\log p+S(p-1))^{O(1)}$, where n is the degree of the polynomial to be factored.*

Remark 1.2. We improve the factoring result of von zur Gathen [1987, Section 4] in that we use an element from $GF(p)$ of order $p_1p_2\dots p_r$ instead of an element of

---

* Research partially supported by Hungarian National Foundation for Scientific Research, Grant 1812.

AMS subject classification (1980): 11 Y 16, 68 Q 25, 68 Q 40

order $p-1$. Putting it another way, for primes with $S(p-1)$ small we can factor polynomials if we can factor the cyclotomic polynomials $\frac{x^{p_i}-1}{x-1}$.

We use a variant of the linear algebraic technique introduced in Rónyai [1987, Section 2]. This approach allows a natural and concise treatment of the problems involved.

Moenck [1977, Theorem 6] presented a deterministic polynomial time method to factor polynomials over $GF(p)$, provided that a primitive element from $GF(p)^\times$ is given and $p-1$ has the form $p-1=2^lL$, where L is odd and $L=O(\log p)$. We offer a stronger and slightly more general result in this direction.

Theorem 1.3. *Suppose that $p-1=t^lT$, t is a prime, $l\geqq 1$ and $gcd(t,T)=1$. Then we can factor polynomials over $GF(p)$ in time $(t+T+n+\log p)^{O(1)}$ where n is the degree of the polynomial to be factored.*

Thus, if t and T are small then we have a polynomial time factoring method without having $\mathrm{soc}(p)$ explicitly given.

Let G_n denote the set of invertible n by n matrices over the field $GF(p)$ which are similar over $GF(p)$ to a diagonal matrix. S_n denotes the set of n by n invertible scalar matrices over $GF(p)$ (i.e. the matrices of form αI where $0\neq\alpha\in GF(p)$ and I is the n by n identity matrix).

The multiplicative group $GF(p)^\times$ of $GF(p)$ can be written as the direct product of its Sylow p_i subgroups P_i

$$GF(p)^\times = P_1\times P_2\times\ldots\times P_r.$$

For an element $\alpha\in GF(p)^\times$ let $o(\alpha)$ denote the multiplicative order of α, i.e. $o(\alpha)$ is the smallest positive integer k such that $\alpha^k=1$. Every element $\alpha\in GF(p)^\times$ can be expressed uniquely as

$$\alpha = \alpha_1\alpha_2\ldots\alpha_r$$

where $\alpha_i\in P_i$. The element α_i can be obtained from α using $(\log p)^{O(1)}$ bit operations if the primes p_i are known. Indeed, we can efficiently compute the multiplicative inverse $1\leqq c_i\leqq p^{e_i}$ of the integer $d_i=\frac{p-1}{p_i^{e_i}}$ modulo p^{e_i}. Next, using fast exponentiation, we can compute $\alpha_i=\alpha^{d_ic_i}$.

This observation is readily generalized to matrices as follows. If $A\in G_n$, then A can be written as

$$(*)\qquad A = A_1A_2\ldots A_r$$

where $A_i=A^{d_ic_i}\in G_n$, the characteristic roots of A_i are in P_i and the relations $AA_i=A_iA$ hold. The matrices A_i can be obtained from A in time $(n+\log p)^{O(1)}$ if the primes p_i are known.

The problem of factoring polynomials over $GF(p)$ is closely related to the problem of finding invariant subspaces of matrices over $GF(p)$. Indeed, if

$$f(x) = a_0+a_1x+\ldots+a_{n-1}x^{n-1}+x^n\in GF(p)[x]$$

is a polynomial to be factored then we can form the *companion matrix* A_f of f

$$A_f = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{pmatrix}.$$

A_f is an n by n matrix over $GF(p)$ and for the characteristic polynomial we have $\det(A_f - xI) = (-1)^n f(x)$. If we have a nontrivial invariant subspace U of A_f (acting on the linear space of column vectors of length n over $GF(p)$), then we can consider the action of A_f on U. In this way we obtain a linear transformation of U, and its characteristic polynomial is a nontrivial divisor of f. To obtain a nontrivial invariant subspace of A_f, we try to find a matrix B such that $BA_f = A_f B$ and the characteristic polynomial f_B of B has at least two different roots in $GF(p)$. If α is a root of f_B then $\ker(B - \alpha I)$ is a nontrivial invariant subspace of A_f. A particularly important special case is when $f_B | x^p - x$ (i.e. f_B splits in $GF(p)$ and has no multiple roots). Then f also splits in $GF(p)$ and from the roots of f_B we can obtain the roots of f because of $\dim_{GF(p)} \ker(B - \alpha I) = 1$ for every root α of f_B. An other useful special case is when $B \in G_n \setminus S_n$ and f_B has multiple roots. Then for the minimal polynomial h of B we have $h | x^{p-1} - 1$ and $\deg h < n$. If we can find a nontrivial factor h_1 of h then we can find a nontrivial factor of f, for $\ker(h_1(B))$ is a nontrivial invariant subspace of the matrix A_f. We call a matrix B a *splitting matrix for* f if $BA = AB$, $B \in G_n \setminus S_n$ and f_B has multiple roots.

We note that the linear algebraic computations mentioned in the preceding discussion (finding characteristic polynomial, minimal polynomial, computing the kernel, computing the action on an invariant subspace) can be done using $(n + \log p)^{O(1)}$ bit operations.

2. Matrices and polynomials

In this section we collect some facts and results related to the factoring algorithm to be presented in Section 3.

Fact 2.1. *Suppose that* $H, K \subset GF(p)$, $|K| + |H| = k \leq p - 2$ *and let* $\alpha \neq \beta \in GF(p)$. *Then there exists an integer* i, $0 \leq i \leq k + 1$ *such that* $i \notin K$ *and* $\frac{\alpha + i}{\beta + i} \notin H$.

Proof. It is clear that for $\gamma \in GF(p)$ the equation

$$\frac{\alpha + x}{\beta + x} = \gamma$$

has at most one solution x_γ. Also, $\beta + x = 0$ holds only for $x = -\beta$. Thus, the number of forbidden values of i is at most $|K| + |H| + 1 \leq k + 1$. The elements i, $0 \leq i \leq k + 1$ are different modulo p, hence the result follows. ∎

Fact 2.2. *Let $\alpha \neq \beta \in GF(p)^\times$. If $\alpha^{p_i} = \beta^{p_i}$ for some $1 \leqq i \leqq r$ then $\alpha/\beta \in \mathrm{soc}\,(p)$.*

Proof. If $\alpha^{p_i} = \beta^{p_i}$ then $(\alpha/\beta)^{p_i} = 1$ hence $\alpha/\beta \in \mathrm{soc}\,(p)$. ∎

For a matrix $A \in G_n$ and a positive integer i we put $A_{(i)} = A + iI$ where I denotes the n by n identity matrix. The decomposition (*) of $A_{(i)}$ is written as

$$A_{(i)} = A_{i1} A_{i2} \dots A_{ir}.$$

Lemma 2.3. *Let $A \in G_n \setminus S_n$ and suppose that $n + |\mathrm{soc}\,(p)| \leqq p-2$. Then there exist integers i, j, $0 \leqq i \leqq m+n+1$, $1 \leqq j \leqq r$ such that one of the following statements holds:*
(a) $n \neq p_j$ $A_{ij} \in G_n \setminus S_n$.
(b) $n = p_j$ *and* $A_{ij}^{p_j} \in G_n \setminus S_n$.

Proof. As $A \in G_n \setminus S_n$, it has at least two different eigenvalues $\alpha, \beta \in GF(p)^\times$. Applying Fact 2.1 with $H = \mathrm{soc}\,(p)$ and $K = \{-\gamma;\ \gamma$ *is a characteristic root of* $A\}$, we obtain that there exists an integer i, $0 \leqq i \leqq m+n+1$ such that $A_{(i)} \in G_n$ and $A_{(i)}$ has two characteristic roots $\gamma, \delta \in GF(p)^\times$ for which $\gamma/\delta \notin \mathrm{soc}\,(p)$. Now we consider the matrices A_{ij}. If at least two of them are in $G_n \setminus S_n$ then (a) holds. We can therefore assume that there exists exactly one j such that $A_{ij} \in G_n \setminus S_n$. If $n \neq p_j$ then we have (a) again. Now suppose that $n = p_j$. Clearly $A_{ij}^{p_j} \in S_n$ if and only if $A_k^{p_j} \in S_n$. By Fact 2.2 the latter is impossible and hence (b) follows. ∎

Remark 2.4. For a given matrix $A \in G_n \setminus S_n$ the matrices A_{ij} can be computed using $(m+n+\log p)^{O(1)}$ bit operations, hence we can efficiently find a matrix A_{ij} satisfying (a) or (b) of Lemma 2.3.

Fact 2.5. *Let t be a prime and $A \in G_t$ such that $A^t = \alpha I \in S_t$. Suppose further that the characteristic polynomial f_A of A has no multiple roots. Then* $\det(A) = (-1)^{t+1}\alpha$.

Proof. If f_A has no multiple roots, then we have $f_A = (-1)^t(x^t - \alpha)$. Using the fact that $\det(A)$ is the constant term of f_A, the statement follows. ∎

Fact 2.6. *Let t be a prime and $A \in G_t$ such that $A^{t^2} = \alpha I \in S_t$. Suppose that A^t has no multiple characteristic roots. Then* $\det(A)^t = (-1)^{t+1}\alpha$. ∎

Proof. Using Fact 2.5, we obtain that $(-1)^{t+1}\alpha = \det(A^t) = \det(A)^t$. ∎

Fact 2.7. *Let $t = p_i$ be an arbitrary prime divisor of $p-1$ and $A \in G_n$, $n < t$ such that $A^t = \alpha I \in S_n$ for $1 \neq \alpha \in P_i$. Then* $\det(A) \in P_i$ *and* $o(\det(A)) > o(\alpha)$.

Proof. From $gcd(n, t) = 1$ we infer $1 < t^k = o(\alpha) = o(\alpha^n)$. Also we have $\alpha^n = \det(A^t) = \det(A)^t$. This implies that $\det(A^{t^{k+1}}) = 1$ hence $\det(A) \in P_i$. Finally, we observe that $\det(A^{t^k}) = 1$ is impossible because it would imply $\alpha^{t^{k-1}} = 1$, a contradiction. ∎

Lemma 2.8. *Let t be a prime dividing $p-1$ and suppose that we have elements $\alpha, \beta \in GF(p)^\times$ such that $o(\alpha) = t^k < o(\beta) = t^l$. Then the roots of the polynomial $x^t - \alpha$ are in $GF(p)$ and can be found in time polynomial in $t + \log p$.*

Proof. The conditions imply that α is in the multiplicative subgroup generated by β. More precisely, there exists a positive integer $j \leqq t^l$ such that $\alpha = \beta^j$ and j is divisible by t. This exponent j can be computed in time polynomial in t and $\log p$ using essentially the Tonelli—Shanks algorithm (Tonelli [1891], Shanks [1972], Adleman, Manders, Miller [1977], Huang [1985, Section 2], von zur Gathen [1987, Lemma 3.1]). Also, using β, we can find a primitive t-th root of unity γ from $GF(p)$ in time $(t+\log p)^{O(1)}$. Now observing that the roots of $x^t-\alpha$ are $\beta^{j/t}\gamma^i$ where $1\leqq i\leqq t$, the statement is proved. ∎

The following statement is similar to Lemma 2.3. We let $t=p_1$ and $T=\frac{p-1}{p_1^{e_1}}$.

Lemma 2.9. *Let $A\in G_n\setminus S_n$ and suppose that $n+tT\leqq p-2$. Then there exists an integer i, $0\leqq i\leqq n+tT+1$ such that $A_{i1}^t\in G_n\setminus S_n$.*

Proof. Let $\alpha,\beta\in GF(p)^\times$ be two different eigenvalues of A. We apply Fact 2.1 with

$$H=\{\delta\in GF(p);\ \delta^{tT}=1\} \tag{1}$$

and

$$K=\{-\gamma;\ \gamma \textit{ is a characteristic root of } A\}.$$

We obtain that there exists an integer i, $0\leqq i\leqq n+tT+1$ such that $A_{(i)}\in G_n$ and $A_{(i)}$ has two nonzero eigenvalues ξ, η such that $\xi/\eta\notin H$. The latter fact implies that $\xi^{tT}\neq\eta^{tT}$ and therefore $A_{(i)}^{tT}\notin S_n$. Also we have $A_{ij}^{tT}=I$ for $j>1$ and thus $A_{(i)}^{tT}=$ $=A_{i1}^{tT}$ and the statement follows. ∎

3. The factoring method

Now we are in the position to describe the factoring procedure of Theorem 1.1. The input is a polynomial $f\in GF(p)[x]$, $\deg(f)=n>1$ such that the roots of f are in $GF(p)$. We shall also assume that $f(0)\neq 0$ and that f has no multiple roots. Algorithm 1 either produces the complete factorization of f or a splitting matrix $B\in G_n$ for f. We assume that $\mathrm{soc}(p)$ is explicitly given. We can also assume that $n+m\leqq p-2$, otherwise one can use Berlekamp's algorithm (Berlekamp [1968], [1970], Knuth [1981], Lidl, Niederreiter [1983]) to obtain the complete factorization of f.

Algorithm 1.

Step 1. *Form the companion matrix $A=A_f\in G_n$ and compute the matrices A_{ij} $0\leqq i\leqq n+m+1$ and $1\leqq j\leqq r$. Find indices i, j for which one of the alternatives of Lemma 2.3 holds and put $B=A_{ij}$, $t=p_j$. Next generate the sequence of matrices*

$$B, B^t, B^{t^2}, \ldots, B^{t^k}$$

until we obtain a scalar matrix $B^{t^k}=\alpha I$. Compute the characteristic polynomial g of $B^{t^{k-1}}$.

Step 2. *If the characteristic polynomial g of $B^{t^{k-1}}$ has multiple roots then $B^{t^{k-1}}$ is a splitting matrix for f,* **return** $(B^{t^{k-1}})$.
(* *The polynomial g has no multiple roots, consequently $n\leqq t$.* *)

Step 3. *If $n\neq t$ then skip the rest of this step.*
(* *For the rest of Step 3 we have $n=t$ and $k\geqq 2$.* *)

If t is odd, then by Fact 2.6 $\det(B^{t^{k-1}})$ *is a root of g, and having* $\mathrm{soc}\,(p)$ *(and thus a primitive t-th root of unity) at hand, we find all the roots of g and then find the complete factorization of f.*
(In the remaining part of Step* 3 *we settle the case* $t=2$. *Observe that* $t=2$ *and* $k\geqq 2$ *imply that* $4|p-1$, *therefore the polynomial* x^2+1 *splits in* $GF(p)$. *)
If $t=2$, *then first we find a root* γ *of the polynomial* x^2+1, *using the algorithm of Schoof* [1985] *for taking modular square roots of small integers. Now* $\gamma\det(B^{t^{k-2}})$ *is a root of g and we proceed as in the odd case to find the roots of f. In all cases we return the complete factorization of f.*

Step 4. *(* Here we have* $n<t$ *and* $k>0$. *)
If $\alpha=1$ *then the roots of g are in* $\mathrm{soc}\,(p)$ *and we find them by computing the elements* $g(\zeta)$, $\zeta\in\mathrm{soc}\,(p)$. *In this way we obtain the complete factorization of f. If* $\alpha\neq 1$ *then by Fact* 2.7 $\det(B^{t^{k-1}})\in P_j$ *and* $o(\det(B^{t^{k-1}}))>o(\alpha)$, *so by Lemma* 2.8 *we can factor* $x^t-\alpha$ *and thus find the roots of g. Again, we obtain the complete factorization of f.*
We return the complete factorization of f.

End

Lemma 3.1. *Let* $f\in GF(p)[x]$ *be a polynomial such that* $\deg f=n>1$, $f|x^{p-1}-1$. *Suppose further that the set* $\mathrm{soc}\,(p)$ *is explicitly given. Then Algorithm* 1 *either finds the roots of f or produces a splitting matrix for f. It runs in deterministic time* $(n+S(p-1)+\log p)^{O(1)}$.

Proof. If we finish at Step 2 then we have a splitting matrix for f. If we finish at Step 3 or at Step 4 then we have found all the roots of f. Observing that $k\leqq\log p$ and $t\leqq S(p-1)$ the timing follows. ▌

Now we can prove Theorem 1.1.

Proof of Theorem 1.1. The problem of factoring $f\in GF(p)[x]$, $\deg(f)=n$ can be reduced in time $(n+\log p)^{O(1)}$ to finding roots in $GF(p)$ of at most n polynomials of degree at most n, using Berlekamp's reduction (Berlekamp [1968], [1970], Knuth [1981], Lidl, Niederreiter [1983]). The polynomials obtained split into linear factors in $GF(p)$. We can assume that $f(0)\neq 0$ and by computing $gcd(f,x^{p-1}-1)$ we can assure that f has no multiple roots. This can be done using $(n+\log p)^{O(1)}$ bit operations.

Clearly it suffices to show that f can be factored into at least two nonconstant factors in time $(n+S(p-1)+\log p)^{O(1)}$. To this end we apply Algorithm 1. If it finds the roots of f then we are done. Otherwise it returns a splitting matrix B_1 for f. Let f_1 denote the minimal polynomial of B_1. We have $f_1|x^{p-1}-1$ and $1<\deg f_1<<\deg f$. We can apply Algorithm 1 to f_1 and so on. More formally we compute a sequence of matrices $B_1,\dots,B_i$ and a sequence of polynomials $f=f_0,f_1,\dots,f_i$ such that B_{j+1} is a splitting matrix for f_j, $0\leqq j<i$ and $f_j|x^{p-1}-1$ is the minimal polynomial of B_j, until f_i is completely factored by our algorithm. Using the fact that $\deg f_j>\deg f_{j+1}>1$, these sequences can be generated in time $(n+S(p-1)+\log p)^{O(1)}$. Then, proceeding backwards, from the factorization of f_i we obtain (partial) factorization of $f_{i-1},\dots,f_0=f$ using repeatedly the technique described at the end of Section 1. This task can be executed in time $(n+\log p)^{O(1)}$. We have obtained that f can be factored into at least two nonconstant factors using $(n+S(p-1)+\log p)^{O(1)}$ bit operations. The proof is complete. ▮

Now we turn to the algorithm of Theorem 1.3. First we recall that $p-1=t^l T$, where t is a prime, $l>0$ and $gcd(t, T)=1$. Let P_1 denote the Sylow t subgroup of $GF(p)^*$. We observe that a primitive t-th root of unity $\eta \in GF(p)^\times$ can be found in time $(\log p+T)^{O(1)}$ by simply examining at most $T+1$ elements of $GF(p)^\times$. Clearly at least one of them has a nontrivial component in P_1 and then by computing a suitable power of such an element we obtain a primitive t-th root of unity.

We can now proceed along the lines of the proof of Theorem 1.1 First we give a variant of Algorithm 1 adapted to the present situation. We impose the same assumptions on the input polynomial f (i.e. $f \in GF(p)[x]$, $\deg(f)=n>1$ and $f|x^{p-1}-1$). The method either produces the complete factorization of f or gives a splitting matrix for f. We can assume that $n+tT \leqq p-2$.

Algorithm 2.

Step 1. *Form the companion matrix* $A=A_f \in G_n$ *and compute the matrices* A_{i1} *for* $0 \leqq i \leqq n+tT+1$. *Find a subscript i for which the statement of Lemma* 2.9 *holds and put* $B=A_{i1}$, $t=p_1$. *Next generate the sequence of matrices*

$$B, B^t, B^{t^2}, \ldots, B^{t^k}$$

until we obtain a scalar matrix $B^{t^k}=\alpha I$. *Compute the characteristic polynomial g of* $B^{t^{k-1}}$.

Steps 2—4. *Identical to the corresponding steps of Algorithm* 1.
(Observe that we need only a* p_1*-th root of unity from* soc(p) *which is available at cost* $(\log p+T)^{O(1)}$. **)*

End

We thus have the following

Lemma 3.2. *Algorithm* 2 *either finds the roots of f or produces a splitting matrix for f. It runs in deterministic time* $(\log p+t+T+n)^{O(1)}$. ▮

Now Theorem 1.3 can be proved similarly to Theorem 1.1, using Algorithm 2 instead of Algorithm 1. ▮

References

[1] L. Adleman, G. Miller and K. Manders, On taking roots in finite fields; *Proc. 18th IEEE Symp. on Foundations of Computer Science*, (1977), 175—178.
[2] E. R. Berlekamp, *Algebraic coding theory;* McGraw-Hill, 1968.
[3] E. R. Berlekamp, Factoring polynomials over large finite fields; *Math. Computation*, **24** (1970), 713—735.
[4] J. von zur Gathen, Factoring polynomials and primitive elements for special primes; *Theoretical Computer Science*, **52** (1987), 77—89.
[5] M. A. Huang, Riemann Hypothesis and finding roots over finite fields; *Proc. 17th ACM Symp. on Theory of Computing*, (1985), 121—130.
[6] D. E. Knuth, *The art of computer programming;* Vol. 2, Seminumerical algorithms Addison-Wesley Publishing Co., 1981.
[7] R. Lidl and H. Niederreiter, *Finite fields;* Addison-Wesley Publishing Co., 1983.

[8] R. T. Moenck, On the efficiency of algorithms for polynomial factoring; *Mathematics of Computation,* **31** (1977), 235—250.
[9] L. Rónyai, Factoring polynomials over finite fields; *Proc. 28th IEEE Symp. on Foundations of Computer Science,* (1987), 132—137.
[10] R. J. Schoof, Elliptic curves over finite fields and the computation of square roots $\bmod p$; *Mathematics of Computation,* **44** (1985), 483—494.
[11] D. Shanks, Five number-theoretic algorithms; in *Proc. 1972 Number Theory Conference, University of Colorado,* Boulder, 1972, 217—224.
[12] A. Tonelli, *Göttinger Nachrichten,* (1891), 344—346. Also in L. E. Dickson, *History of the theory of numbers,* Chelsea, New York, Vol. **I,** 215.

Lajos Rónyai

Computer and Automation Institute
Hungarian Academy of Sciences
Budapest, P.O.B. 63
H—1502 Hungary