Contemporary Mathematics
Volume **168**, 1994

Berlekamp's and Niederreiter's Polynomial Factorization Algorithms

SHUHONG GAO AND JOACHIM VON ZUR GATHEN

ABSTRACT. In this paper, we discuss Niederreiter's algorithm for factoring polynomials over finite fields and compare it to Berlekamp's algorithm. Using Kaltofen and Saunders's version of Wiedemann's method for solving systems of linear equations, we present a probabilistic factorization algorithm. This approach is particularly interesting in characteristic two, where it seems somewhat faster than Kaltofen's adaptation of Berlekamp's factorization algorithm.

1. Introduction

Niederreiter [**30**] bases his algorithm for factoring a polynomial $f \in \mathbb{F}_q[x]$ on the differential equation

$$f^p (h/f)^{(p-1)} + h^p = 0 \tag{1.1}$$

where $h \in \mathbb{F}_q[x]$ with $\deg h < \deg f = n$ is unknown, $p = \operatorname{char} \mathbb{F}_q$, and $(h/f)^{(p-1)} = \partial^{p-1}(h/f)/\partial x^{p-1}$ is the derivative of order $p-1$. This corresponds to a system of n linear equations in the n unknown coefficients of h; the only nonzero coefficients in either summand occur at powers x^i with i divisible by p. A variant of this method works directly over $\mathbb{F}_q$, replacing the ordinary (formal) derivative in (1.1) by the Hasse-Teichmüller derivative. If q is a power of two, the resulting system of linear equations over $\mathbb{F}_q$ has a particularly simple format. The current status of this approach and further references are given in [**29**].

1991 *Mathematics Subject Classification.* Primary 11T06, 68Q40; Secondary 11Y16, 12Y05, 68Q25.

Parts of this work were done during a sabbatical visit by the second author to the Institute for Scientific Computation at ETH Zürich, whose hospitality is gratefully acknowledged. The research was supported by the Information Technology Research Centre of Ontario, and the Natural Sciences and Engineering Research Council of Canada. The first author was supported by an NSERC post doctoral fellowship.

This paper is in final form, no version of it will be submitted for publication elsewhere.

In Section 2, we discuss a simple relation between Niederreiter's and Berlekamp's algorithms [**2, 3**], discovered by Fleischmann [**10**], Lee & Vanstone [**23**], and Miller [**25**]. The novelty is that we turn this relation around and use it to obtain a short self-contained proof of all the salient features of Niederreiter's method.

There are several extensions of Niederreiter's basic algorithm, via Hasse-Teichmüller derivatives and normal bases. Such extensions are desirable for a better understanding of the scope of the method. In this paper we only discuss the basic method. Niederreiter's algorithm has found application in other areas such as characteristic sequences [**32**]. We do not discuss these applications at all.

Niederreiter was exclusively interested in deterministic algorithms. In this paper, we present a probabilistic algorithm based on his approach, using Kaltofen & Saunders's randomized algorithm [**21**] founded on Wiedemann's method [**41**], which we apply in Section 3 to solve the linear equations (1.1). An important tool in some factoring algorithms is the "$(p-1)/2$ power method"; we show how to apply it in the context of Niederreiter's approach.

Our algorithm is particularly interesting in the case $p = 2$, which we discuss in Section 4. Then the equation (1.1) takes an especially simple form, and the well-known open problem of even obtaining deterministic polynomial time in large characteristic does not arise in this situation. Now the obstacle to deterministic algorithms is that at problem sizes near the edge of current feasibility the "explicit" way of doing linear algebra, where the matrix of a system of linear equations is stored explicitly (in dense or sparse format), has to be replaced by the blockbuster method of Wiedemann. Our algorithm gives a solution to Problem 1 in [**28**], and seems somewhat faster than Kaltofen's version [**18**, p. 297] of Berlekamp's algorithm (using about half the time, for a large enough field).

Until recently, the best known factoring algorithms required time either about n^ω or about $n^2 \log q$, ignoring factors $\log n$, and where ω is an exponent for matrix multiplication. Kaltofen [**18**] and von zur Gathen & Shoup [**15**] improved this to about $n^2 + n \log q$; the algorithm of Section 4 achieves a similar time bound in characteristic two.

Since this paper addresses a rather wide, mainly mathematical, audience, we have considered it useful to include in Section 5 our personal view on algorithm design and efficiency—a view widely shared in Computer Science.

2. Niederreiter's method

In this section, we give a short stand-alone derivation of those well-known features of Niederreiter's algorithm that seem to be necessary for the goal of an efficient factorization algorithm. In the first part, we only discuss squarefree polynomials; this may be sufficient for efficient algorithms (see Section 5). For the special case $q = 2$, the proof can be further simplified [**29**].

Let $f \in \mathbb{F}_q[x]$ be a monic squarefree polynomial of degree n, whose monic irreducible factors $g_1, \ldots, g_r \in \mathbb{F}_q[x]$ we want to compute. Let $p = \operatorname{char} \mathbb{F}_q$, and denote by $\mathcal{B}' \subseteq \mathbb{F}_q[x]$ the algebra

$$\mathcal{B}' = \{h \in \mathbb{F}_q[x] : h^p \equiv h \bmod f\},$$

and by $\mathcal{B} \subseteq \mathcal{R}$ its image in

$$\mathcal{R} = \mathbb{F}_q[x]/(f).$$

This is the absolute *Berlekamp algebra*, an r-dimensional vector space over $\mathbb{F}_p$.

Niederreiter considers

$$\mathcal{N} = \left\{h \in \mathbb{F}_q[x] : \left(\frac{h}{f}\right)^{(p-1)} + \left(\frac{h}{f}\right)^p = 0\right\}, \tag{2.1}$$

where $(h/f)^{(p-1)} = \partial^{p-1}(h/f)/\partial x^{p-1}$ is the derivative of order $p-1$. Since both operations of taking derivatives and pth powers are $\mathbb{F}_p$-linear in the coefficients of h, $\mathcal{N} \subseteq \mathbb{F}_q[x]$ is an $\mathbb{F}_p$-vector space. The following map, discovered by Fleischmann [**10**], Lee & Vanstone [**23**], and Miller [**25**], provides a connection between $\mathcal{B}$ and $\mathcal{N}$:

$$\begin{array}{rccc} \psi\colon & \mathbb{F}_q[x] & \to & \mathbb{F}_q[x], \\ & h & \mapsto & hf' \operatorname{rem} f, \end{array}$$

where $hf' \operatorname{rem} f$ denotes the remainder modulo f, i.e., the unique polynomial of degree less than n congruent to hf' modulo f. Statement (i) of the following theorem is in [**10, 23, 25**], (ii) is in [**30**], and (iii) is in [**30**] for $a = 0$ and in [**33**, Theorem 2] in a more complicated form. Niederreiter & Göttfert [**32**] show that the matrices describing Niederreiter's and Berlekamp's methods are similar; this is also a consequence of Theorem 2.1. (iv) and (v) seem to be new, as is our approach of deriving information on $\mathcal{N}$ from $\mathcal{B}$. Although the property of being an algebra gets lost in the transition from $\mathcal{B}$ to $\mathcal{N}$, (v) shows that one can use the well-known tool (for probabilistic algorithms) of $(p-1)/2$-th powers also directly in $\mathcal{N}$.

THEOREM 2.1. *Let $f \in \mathbb{F}_q[x]$ be squarefree.*

(i) *The restriction of ψ to $\mathcal{B}'$ is a surjective $\mathbb{F}_p$-linear homomorphism onto $\mathcal{N}$ and induces an isomorphism $\overline{\psi} : \mathcal{B} \to \mathcal{N}$.*

(ii) $\dim_{\mathbb{F}_p} \mathcal{N} = r$, *and* $\{fg_i'/g_i : 1 \le i \le r\}$ *is a basis for $\mathcal{N}$ over $\mathbb{F}_p$.*

(iii) *If $a \in \mathbb{F}_p$ and $h = \sum_{1 \le i \le r} a_i f g_i'/g_i \in \mathcal{N}$, with $a_1, \ldots, a_r \in \mathbb{F}_p$, then*

$$\gcd(f, h - af') = \prod_{\substack{1 \le i \le r \\ a_i = a}} g_i.$$

(iv) *For any $h \in \mathcal{N}$, we have*

$$f = \prod_{a \in \mathbb{F}_p} \gcd(f, h - af').$$

(v) *If p is odd, $h \in \mathcal{N}$ as in (iii), and $S \subseteq \mathbb{F}_p$ the set of nonzero squares, then*

$$\gcd(f, h^{(p-1)/2} - (f')^{(p-1)/2}) = \prod_{\substack{1 \le i \le r \\ a_i \in S}} g_i.$$

Before proving Theorem 2.1, we collect some facts about derivatives. For $f, h \in \mathbb{F}_q[x]$ with $f \neq 0$, we write

$$\deg\Big(\frac{h}{f}\Big) = \deg h - \deg f.$$

In Lemma 2.1 below, statements (i)—(iii) are from [**30**], and (iv) is in [**10**].

LEMMA 2.1. *Let f be as above, $h \in \mathbb{F}_q[x]$, $k \in \mathbb{N}$, and $d_k = (h/f)^{(k)} f^{k+1}$. Then*

(i) $(f'/f)^{(p-1)} + (f'/f)^p = 0$,
(ii) $\deg\big((h/f)^{(k)}\big) \le \deg(h/f) - k$,
(iii) *for every $h \in \mathcal{N}$, $\deg h < n$,*
(iv) $d_k \in \mathbb{F}_q[x]$, *and* $d_k \equiv (-1)^k k!\, h(f')^k \bmod f$.

PROOF. For (i), note that f has no repeated roots. Let W be the set of roots of f in some extension field of $\mathbb{F}_q$. Then

$$\begin{aligned}\Big(\frac{f'}{f}\Big)^{(p-1)} &= \Big(\sum_{\alpha \in W} \frac{1}{x-\alpha}\Big)^{(p-1)} = \sum_{\alpha \in W} \Big(\frac{1}{x-\alpha}\Big)^{(p-1)} \\ &= (-1)^{p-1}(p-1)! \sum_{\alpha \in W} \frac{1}{(x-\alpha)^p} = -\Big(\frac{f'}{f}\Big)^p.\end{aligned}$$

(ii) is a straightforward proof by induction on k, and (iii) follows from (ii) by comparing degrees in (1.1). For (iv), one can easily see that $d_k \in \mathbb{F}_q[x]$, and the congruence follows inductively from

$$d_k f' \equiv d_k f' + d_k' f \equiv (d_k f)' \equiv d_{k+1} + (k+2) d_k f' \bmod f. \qquad \square$$

PROOF OF THEOREM 2.1. (i) We first show that $\psi(\mathcal{B}') \subseteq \mathcal{N}$. So let $g \in \mathcal{B}'$, $h = \psi(g)$, and let $u, v \in \mathbb{F}_q[x]$ such that

$$g = g_*^p + fu, \quad \text{and} \quad gf' = h + fv.$$

Then

$$h = g^p f' + f(f'u - v).$$

Let $w = f^p (h/f)^{(p-1)}$. Then $w \in \mathbb{F}_q[x]$, and

$$w \equiv f^p \Big(g^p \frac{f'}{f}\Big)^{(p-1)} \equiv f^p g^p \Big(\frac{f'}{f}\Big)^{(p-1)} \equiv -g^p (f')^p \equiv -h^p \bmod f^p, \tag{2.2}$$

where we have used the Leibniz rule, the fact that $(g^p)^{(i)} = 0$ for $i \geq 1$ in the second equation, and Lemma 2.1 (i) in the third equation. Now $\deg(h/f) \leq -1$, and

$$\deg w = np + \deg\Big((h/f)^{(p-1)}\Big) \leq np - p,$$

by Lemma 2.1 (ii). Furthermore, $\deg h^p \leq (n-1)p$ and $\deg f^p = np$, so that (2.2) implies that $w = -h^p$ and $h \in \mathcal{N}$.

To show that ψ is surjective onto $\mathcal{N}$, let $s \in \mathbb{F}_q[x]$ be such that $sf' \equiv 1 \bmod f$, $h \in \mathcal{N}$, and $g = hs$. Then $h = \psi(g)$, and

$$\begin{aligned} g^p \equiv (hs)^p &\equiv -(h/f)^{(p-1)} f^p s^p \equiv -(-1)^{p-1}(p-1)!\, h(f')^{p-1} s^p \\ &\equiv h(sf')^{p-1} s \equiv hs \equiv g \bmod f, \end{aligned}$$

by Lemma 2.1 (iv). Hence $g \in \mathcal{B}'$ and $h \in \psi(\mathcal{B})$. Furthermore, ψ gives a bijection from $\mathcal{B}$ to $\mathcal{N}$, since $\psi(g) = 0$ if and only if $g \equiv 0 \bmod f$.

(ii) For $1 \leq i \leq r$, let s be as above and

$$e_i = (s\frac{f}{g_i} g_i' \bmod f) \in \mathcal{B}.$$

Then $e_1, \ldots, e_r$ are the primitive idempotents in $\mathcal{B}$, that is, $e_i \equiv \delta_{ij} \bmod g_j$ for all $i, j \leq r$, where δ_{ij} is the Kronecker symbol. Since we know, say from the Chinese remainder decomposition of $\mathbb{F}_q[x]/(f)$, that $\{e_1, \ldots, e_r\}$ is a basis for $\mathcal{B}$ over $\mathbb{F}_p$, it follows from (i) that

$$\{\psi(e_i) : 1 \leq i \leq r\} = \{\frac{f}{g_i} g_i' : 1 \leq i \leq r\}$$

is a basis for $\mathcal{N}$.

(iii) follows from the facts that $f' = \sum_{1 \leq i \leq r} f g_i'/g_i$ and that $g_i | (h - af')$ if and only if $a_i = a$, and (iv) follows from (iii).

For (v), since $(fg_i'/g_i) \cdot (fg_j'/g_j) \equiv 0 \bmod f$ if $i \neq j$, we have

$$h^k \equiv \sum_{1 \leq i \leq r} a_i^k (fg_i'/g_i)^k \bmod f,$$

for any positive integer k. Thus

$$h^{(p-1)/2} - (f')^{(p-1)/2} \equiv \sum_{1 \leq i \leq r} (a_i^{(p-1)/2} - 1)(fg_i'/g_i)^{(p-1)/2} \bmod f,$$

and so the left hand side is divisible by g_i if and only if $a_i \in S$. □

If $I \subseteq \{1, \ldots, r\}$, $h = \sum_{i \in I} fg_i'/g_i$ and $b = \prod_{i \in I} g_i$, then $h = fb'/b$. If $p = 2$, then only $a_i \in \{0, 1\}$ occur in the representation of Theorem 2.1 (iii), and the above implies that

$$\mathcal{N} = \{fb'/b : b \in \mathbb{F}_q[x] \text{ monic}, b \,|\, f\},$$

which is Theorem 1 in [**28**].

Niederreiter proves more general versions of some statements in Theorem 2.1, where the subfield $\mathbb{F}_p \subseteq \mathbb{F}_q$ used in (1.1) can be replaced by an arbitrary subfield

$\mathbb{F}_t$ of $\mathbb{F}_q$; in this case the derivative $(h/f)^{(p-1)}$ has to be replaced by the Hasse-Teichmüller derivative $H^{(t-1)}(h/f)$, or one can use a normal basis of $\mathbb{F}_q$ over $\mathbb{F}_t$. This is of interest for a most general statement, and possibly also in applications such as linearly recurrent sequences. The Hasse-Teichmüller approach seems to lead to more complicated and less efficient algorithms.

We now adapt the above results to polynomials that are not squarefree. Let

$$f = g_1^{e_1} \cdots g_r^{e_r},$$

where $g_1, \ldots, g_r$ are the distinct monic irreducible factors of f, and $e_1, \ldots e_r \in \mathbb{N}$ positive. Let $\mathcal{N}$ be as in (2.1), $f_\boxtimes = g_1 \cdots g_r$ the squarefree part of f,

$$\mathcal{N}_\boxtimes = \left\{ g \in \mathbb{F}_q[x] : \left(\frac{g}{f_\boxtimes}\right)^{(p-1)} + \left(\frac{g}{f_\boxtimes}\right)^p = 0 \right\}$$

the space corresponding to $f_\boxtimes$, and

$$\begin{array}{rccc} \varphi\colon & \mathcal{N}_\boxtimes & \to & \mathbb{F}_q[x] \\ & g & \mapsto & g\dfrac{f}{f_\boxtimes}. \end{array}$$

THEOREM 2.2. *φ is an $\mathbb{F}_p$-linear isomorphism between $\mathcal{N}_\boxtimes$ and $\mathcal{N}$.*

PROOF. Obviously $\varphi(g) \in \mathcal{N}$ for all $g \in \mathcal{N}_\boxtimes$, and φ is injective. We claim that $f/f_\boxtimes$ divides each $h \in \mathcal{N}$, from which surjectivity of φ follows easily. Let $h \in \mathcal{N}$,

$$v = \gcd(f, h) = \prod_{1 \le i \le r} g_i^{c_i},$$

with $0 \le c_i \le e_i$ for $1 \le i \le r$, and $w = f_\boxtimes \cdot (f/v)'/(f/v)$. Then

$$w = f_\boxtimes \sum_{1 \le i \le r} \frac{(g_i^{e_i - c_i})'}{g_i^{e_i - c_i}} = f_\boxtimes \sum_{1 \le i \le r} (e_i - c_i)\frac{g_i'}{g_i} \in \mathbb{F}_q[x].$$

For $k \in \mathbb{N}$, let $d_k = f_\boxtimes^k (f/v)(h/f)^{(k)}$. One sees that $d_k \in \mathbb{F}_q[x]$ by induction on k, using

$$\begin{aligned} (d_k f_\boxtimes)' &= (k+1) d_k f_\boxtimes' + f_\boxtimes^{k+1} \left(\frac{f}{v}\right)' \left(\frac{h}{f}\right)^{(k)} + d_{k+1} \\ &= (k+1) d_k f_\boxtimes' + w d_k + d_{k+1}. \end{aligned}$$

Dividing (1.1) by v^p, we find

$$-\left(\frac{h}{v}\right)^p = \left(\frac{f}{f_\boxtimes v}\right)^{p-1} f_\boxtimes^{p-1} \frac{f}{v} \left(\frac{h}{f}\right)^{(p-1)} = \left(\frac{f}{f_\boxtimes v}\right)^{p-1} d_{p-1} \in \mathbb{F}_q[x].$$

For our claim, it is sufficient to prove that $c_i \ge e_i - 1$ for all i. So we assume that $c_i \le e_i - 2$, for some $i \le r$. Then g_i does not divide the left hand side, and its multiplicity in the right hand side is at least

$$(p-1)(e_i - 1 - c_i) \ge p - 1 \ge 1,$$

a contradiction. □

Now consider

$$\begin{aligned}\psi\colon\ \mathbb{F}_q[x] &\to \mathbb{F}_q[x]\\ g &\mapsto gf'_{\boxtimes}\cdot\frac{f}{f_{\boxtimes}} \text{ rem } f,\end{aligned}$$

where $f_{\boxtimes}$ is the squarefree part of f as above. Then Theorems 2.1 and 2.2 imply the following.

COROLLARY 2.1. *In the above notation, statements (i) and (ii) of Theorem* 2.1 *hold for any polynomial* $f \in \mathbb{F}_q[x]$. *Furthermore:*

(iii)' *If* $a \in \mathbb{F}_p$ *and* $h = \sum_{1\le i\le r} a_i f g'_i/g_i \in \mathcal{N}$ *with* $a_1, \dots, a_r \in \mathbb{F}_p$, *then*

$$\gcd(f, h - af') = \frac{f}{f_{\boxtimes}} \prod_{\substack{1\le i\le r\\ a_i = e_i a}} g_i, \quad \gcd\Big(f, h - a\frac{f}{f_{\boxtimes}}f'_{\boxtimes}\Big) = \frac{f}{f_{\boxtimes}} \prod_{\substack{1\le i\le r\\ a_i = a}} g_i.$$

(iv)' *If* $h \in \mathcal{N}$ *is as in (iii)', then*

$$f = \frac{f}{f_{\boxtimes}} w_1 \prod_{a\in\mathbb{F}_p} \frac{f_{\boxtimes}\gcd(f, h - af')}{w_2 f} = \frac{f}{f_{\boxtimes}} \prod_{a\in\mathbb{F}_p} \frac{f_{\boxtimes}}{f} \gcd(f, h - af f'_{\boxtimes}/f_{\boxtimes}),$$

where w_1 *is the product of all* g_i *with* $p \mid e_i$, *and* w_2 *is the product of those* g_i *with* $p \mid e_i$ *and* $a_i = 0$.

PROOF. We just need to prove the first equations in (iii)' and (iv)', since the others follow immediately from the isomorphism. Note that

$$f' = \sum_{1\le i\le r} e_i \frac{f}{g_i} g'_i, \quad h - af' = \frac{f}{f_{\boxtimes}} \sum_{1\le i\le r} (a_i - e_i a)\frac{f_{\boxtimes}}{g_i} g'_i.$$

The first equation in (iii)' follows from the fact that g_k divides $\sum_{1\le i\le r}(a_i - e_i a) f_{\boxtimes} g'_i/g_i$ if and only if $a_k = e_k a$ in $\mathbb{F}_q$, for $1 \le k \le r$. To prove the first equation in (iv)', let $S = \{1, \dots, r\}$, $T = \{i \in S\colon p \mid e_i\}$, and for $a \in \mathbb{F}_p$,

$$U_a = \{i \in S\colon p \nmid e_i \text{ and } a_i = e_i a \in \mathbb{F}_p\}.$$

Then

$$T \cup \bigcup_{a\in\mathbb{F}_p} U_a = S$$

is a partition of S, and for all $a \in \mathbb{F}_p$

$$w_2 \prod_{i\in U_a} g_i = \frac{1}{f} f_{\boxtimes} \gcd(f, h - af'),$$

$$f_{\boxtimes} = \prod_{i\in S} g_i = \prod_{i\in S} g_i \cdot \prod_{a\in\mathbb{F}_p} \prod_{i\in U_a} g_i = w_1 \prod_{a\in\mathbb{F}_p} \frac{f_{\boxtimes}\gcd(f, h - af')}{w_2 f}. \qquad \square$$

We have included the factorizations involving f', since f' is easier to compute than $(f/f_{\boxtimes})f'_{\boxtimes}$; it is not clear that this gains efficiency.

3. Wiedemann's and Niederreiter's methods

We want to factor a squarefree polynomial

$$f = x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in \mathbb{F}_q[x],$$

where $p = \operatorname{char} \mathbb{F}_q$, $q = p^k$ and $f_{n-1}, \ldots, f_0 \in \mathbb{F}_q$. We assume that $\mathbb{F}_q$ is presented by a monic irreducible polynomial of degree k over $\mathbb{F}_p$.

Let $\mathsf{M}(n)$ be such that the product of any two polynomials in $\mathbb{F}_q[x]$ of degrees at most n can be computed with $O(\mathsf{M}(n))$ operations in $\mathbb{F}_q$. We can choose $\mathsf{M}(n) = n \log n \operatorname{loglog} n$ [**36, 5**]. The algorithms of Kaltofen & Saunders [**21**], based on Wiedemann's method [**41**], produce a random vector in the nullspace of a matrix, using $O(n)$ multiplications of the matrix with vectors, plus $O(n\mathsf{M}(n))$ arithmetic operations. To make their probabilistic analysis work, they have to assume that the field contains sufficiently many elements, say at least $50n^2 \log n$ many. If $\mathbb{F}_p$ has too few elements, one has to work over an algebraic field extension of degree $e = \lceil \log_p(50n^2 \log n) \rceil$; this multiplies the running times by a factor

$$E = \begin{cases} \mathsf{M}(e) & \text{if } e \geq 2, \\ 1 & \text{if } e = 1. \end{cases} \tag{3.1}$$

For our situation, let A and B in $\mathbb{F}_p^{nk \times nk}$ be the matrices of the first and second summands of (1.1), respectively, and identify a vector $\vec{h} \in \mathbb{F}_p^{nk}$ with a polynomial $h \in \mathbb{F}_q[x]$ of degree less than n, by presenting $\mathbb{F}_q = \mathbb{F}_{p^k}$ by a monic irreducible polynomial over $\mathbb{F}_p$ of degree k. Then the entries of $A\vec{h}$ and $B\vec{h}$ can be identified with those of $f^p(h/f)^{(p-1)}$ and h^p, respectively. The latter can be calculated by computing n pth powers in $\mathbb{F}_q$, with $n \log p\, \mathsf{M}(k)$ operations in $\mathbb{F}_p$. (It might also be interesting to choose a normal basis of $\mathbb{F}_q$ over $\mathbb{F}_p$.)

To be precise, we have taken the map $h \mapsto h^p$ from the set $\mathcal{F}$ of polynomials with degree less than n to the set $\mathcal{G}$ of polynomials g of degree less than pn with $g' = 0$, and composed it with the linear map given by $x \mapsto x^{1/p}$ from $\mathcal{G}$ to $\mathcal{F}$; similarly for the other map. Then $A + B$ describes a mapping from $\mathcal{F}$ to $\mathcal{F}$ in the standard basis, and $\mathcal{N}$ is its nullspace.

We write $\mathsf{N}(n, p, k)$ for the number of operations in $\mathbb{F}_p$ required to calculate $A\vec{h}$. One way to do this is to calculate A explicitly, and then apply it to $\vec{h}$. According to Niederreiter [**31**, Theorem 1], this can be done with $O(n^\omega \mathsf{M}(k))$ operations in $\mathbb{F}_p$, where $\omega < 2.376$ is an exponent for matrix multiplication, and we assume that $2 < \omega$ for simplicity. This method involves, besides other operations, the calculation of the Berlekamp matrix. The method we now describe is of interest only if $\mathsf{N}(n, p, k)$ is much smaller than above, say roughly linear in n. This is the case for $p = 2$, as explained in Section 4.

A single application of Kaltofen & Saunders's method [**21**] can be done with $O\big(nk \cdot E \cdot (\mathsf{N}(n, p, k) + \mathsf{M}(nk) + n \log p\, \mathsf{M}(k))\big)$ operations in $\mathbb{F}_p$. It returns a

random element of $\mathcal{N}$, i.e.,

$$u_a = \sum_{1 \le i \le r} a_i \frac{f}{g_i} g_i' \in \mathbb{F}_q[x]$$

with $a = (a_1, \ldots, a_r) \in \mathbb{F}_p^r$ random. When $p = 2$, let

$$v_a = \gcd(f, u_a) = \prod_{\substack{1 \le i \le r \\ a_i = 0}} g_i.$$

This is a proper factor of f if and only if $m = \#\{i : a_i = 0\}$ satisfies $0 < m < r$. This occurs with probability $1 - 2^{-(r-1)}$. When p is odd, let

$$v_a = \gcd(f, u_a^{(p-1)/2} - (f')^{(p-1)/2}).$$

By Theorem 2.1 (v), this is a proper factor of f if and only if $m = \#\{i : a_i \in S\}$ satisfies $0 < m < r$. This occurs with probability

$$1 - 2^{-(r-1)}(1 - p^{-1})^r - p^{-r},$$

which is about $1 - 2^{-r+1}$ when p is large. The $(p-1)/2$-th powers modulo f can be computed in $O(\mathsf{M}(n) \log p)$ operations in $\mathbb{F}_q$ or $O(\mathsf{M}(n) \log p\, \mathsf{M}(k))$ operations in $\mathbb{F}_p$. The calculation of the gcd can be done with $O(\mathsf{M}(n) \log n)$ operations in $\mathbb{F}_q$, or $O(\mathsf{M}(n) \log n\, \mathsf{M}(k) \log k)$ operations in $\mathbb{F}_p$. We use $nk\mathsf{M}(nk)$ as an upper bound for the latter.

THEOREM 3.1. *Let p be prime, $k \in \mathbb{N}$, $q = p^k$ and $f \in \mathbb{F}_q[x]$ of degree n. Then the above method finds a proper factor of f with probability at least $1/2$, using an expected number of*

$$O(nk \cdot E \cdot (N(n,p,k) + M(nk) + n \log p\, M(k)))$$

operations in $\mathbb{F}_p$, where E is defined in (3.1).

For implementations, the substantial reduction of the space requirement from n^2 to $O(n)$ is probably more important than the reduction of the number of operations.

For small problems, when efficiency is not an essential concern, one may want to compute and store explicitly the matrices B and N in $\mathbb{F}_p^{nk \times nk}$ describing $\mathcal{B}$ and $\mathcal{N}$, respectively. In the case $p = 2$, no computation is necessary for N, and $O(n\mathsf{M}(n))$ operations in $\mathbb{F}_q$ are sufficient for B. In both cases, one then computes a basis for the solution space; the time for this phase will dominate the total cost. Just as every basis for $\mathcal{B}$ is a *separating set* in the sense of Camion [**4**], Shoup [**37**], and von zur Gathen & Shoup [**15**, Section 9], each basis of $\mathcal{N}$ is a separating set in the sense that each pair (g_i, g_j) of irreducible factors, with $i \ne j$, is "separated" by some element h of the basis (i. e., g_i divides $h - af'$ and g_j does not, for some $a \in \mathbb{F}_q$), since the transition matrix to the basis of Theorem 2.1 (ii) is nonsingular and thus does not have two identical columns. Then the usual procedure of refining partial factorizations by successive gcd's with $h - a$

and $h - af'$, respectively, yields a complete factorization. A somewhat more complicated version of this argument and algorithm is given in [**33, 16**], yielding a total of $O(nk^3 + (nk)^\omega + r^2\mathsf{M}(n)\log n\mathsf{M}(k)\log k)$ operations in $\mathbb{F}_2$, where ω is an exponent of matrix multiplication. The running time of both deterministic algorithms is proportional to p, and can be reduced to $O(\sqrt{p})$ ([**37**] and [**39**, §1.1] by computing the gcd of f with $(h-a)^{(p-1)/2}-1$ and $(h-af')^{(p-1)/2}-(f')^{(p-1)/2}$, respectively.

Note also that our probabilistic approach reduces the number of elements of $\mathcal{N}$ required from r to only $2\log r$ (random elements). This advantage is reflected in the analysis of the running time for complete factorization, which is $\log r$ times the estimate in Theorem 3.1.

4. Characteristic two

The algorithm is particularly simple in characteristic two, where $q = 2^k$ for some $k \in \mathbb{N}$. Equation (1.1) simplifies to

$$(fh)' + h^2 = 0. \tag{4.1}$$

Now $B\vec{h}$ can be computed with n squarings in $\mathbb{F}_q$. For the first summand $A\vec{h}$, we write

$$\begin{aligned} f^+ &= (f + xf')(\sqrt{x}) = \sum_{\substack{i\geq 0\\ i \text{ even}}} f_i x^{i/2} \in \mathbb{F}_q[x], \\ f^- &= (f')(\sqrt{x}) = \sum_{\substack{i\geq 1\\ i \text{ odd}}} f_i x^{(i-1)/2} \in \mathbb{F}_q[x], \end{aligned}$$

for the "contracted" even and odd parts of f, and similarly h^+, h^-. Then $f = f^+(x^2) + xf^-(x^2)$, $f' = f^-(x^2)$, and

$$\begin{aligned} (fh)' &= (f^+(x^2) + xf^-(x^2)) \cdot h^-(x^2) + f^-(x^2) \cdot (h^+(x^2) + xh^-(x^2)) \\ &= (f^+h^- + f^-h^+)(x^2). \end{aligned} \tag{4.2}$$

As Niederreiter [**30**] points out, there is no set-up cost for the matrix A, if one wants it explicitly. From our point of view, $(fh)'$ can be calculated by two multiplications of polynomials of degree at most $n/2$, and the linear operator in (4.1) can be applied to an h with $O(\mathsf{M}(n))$ operations in $\mathbb{F}_q$, or $O(\mathsf{M}(nk))$ operations in $\mathbb{F}_2$ [**15**, Lemma 2.2].

Furthermore, v_a is the product of some g_i's, where each g_i is included in the product with probability $1/2$. Thus v_a is a "random factor" of f, and the standard method of producing finer and finer partial factorization by repeated choice of random factors (see e.g., von zur Gathen & Shoup [**15**]) shows that with at most $2\log r$ applications one can expect to factor f completely. We therefore have the following result.

THEOREM 4.1. *Let $k \in \mathbb{N}$, $q = 2^k$ and $f \in \mathbb{F}_q[x]$ of degree n. Then the above method factors f completely with an expected number of $O(nkM(nk)\log r \cdot E)$ operations in $\mathbb{F}_2$, where $r \le n$ is the number of distinct monic irreducible factors of f, and E is defined in (3.1) with $p = 2$.*

Both in our and Kaltofen's method, the dominant cost is that of applying the matrix to various vectors. For a single application, both methods need n squarings in $\mathbb{F}_q$—at no cost if $q = 2$—plus two multiplications of degree $n/2$ in our case, and one division with remainder by f in the Berlekamp-Kaltofen approach. The former cost is lower than the latter, by a factor of about two in fast arithmetic.

The so-called "classical arithmetic" assumes $\mathsf{M}(n) = n^2$; it is an intuitive model, but does not allow a rigorous definition. In this model, a gcd calculation on polynomials of degree at most n takes $O(n^2)$ operations, and the present factorization method in $\mathbb{F}_{2^k}[x]$ an expected number of $O(n^3k^3 \log r \cdot E)$ operations in $\mathbb{F}_2$. If f is a "sparse" polynomial with at most $t \ll n$ nonzero coefficients, then each $(fh)' - h^2$ can be calculated with $O(nt)$ operations in $\mathbb{F}_{2^k}$. Then we have to compute the remainder of a polynomial of degree less than $2n$ modulo f; this can also be done with $O(tn)$ operations. This leads to a complete factorization algorithm with $O(n^2tk^3 \log r \cdot E)$ operations in $\mathbb{F}_2$. This seems to be the first factorization algorithm that has smaller asymptotic running time for sparse ($t = o(n)$) polynomials than for arbitrary polynomials. Berlekamp's algorithm enjoys the same advantage in this situation. Of course, this is mainly of interest when an $O(nt)$ multiplication algorithm is faster than an FFT-based multiplication, which seems to restrict it to rather small inputs [**38**]. The possibility of exploiting sparseness was observed by Niederreiter & Göttfert [**33**], in the framework of fast explicit linear algebra, where however no specific asymptotic savings can be proved in our situation.

REMARK 4.1. *The trick of (4.2) can be generalized. For $p = 3$, we write $h = h_0(x^3) + xh_1(x^3) + x^2h_2(x^3)$, and similarly for f. Then*

$$f^3(h/f)'' = (f_0f_2 + 2f_1^2)h_0 + (f_0f_1 + 2x^3f_2^2)h_1 + (x^3f_1f_2 + 2f_0^2)h_2.$$

For $p = 5$, the corresponding expression has 70 terms.

REMARK 4.2. *The matrix-vector product corresponding to the nonzero terms in $(fh)'$ is*

$$\begin{pmatrix} f_n & 0 & 0 & 0 & 0 & \cdots \\ f_{n-2} & f_{n-1} & f_n & 0 & 0 & \cdots \\ f_{n-4} & f_{n-3} & f_{n-2} & f_{n-1} & f_n & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{F}_q^{n \times n}.$$

After reordering the columns, this becomes the Sylvester matrix of the contracted even and odd parts of f; this is also obvious from (4.2). It would be interesting to see whether subresultant techniques can be applied to this situation.

5. Conclusion

In the design of polynomial factorization algorithms over finite fields there are several legitimate goals, each of which leads to a yardstick for comparison of different algorithms. A basic distinction is between

1. theoretical, and
2. practical

results. For "practical" results, one can aim at

2.a. extending the range of currently feasible problem sizes,

2.b. new algorithms for the currently feasible range.

Clearly, 2.a is more interesting than 2.b. "Practical" claims will often be supported by actual implementations; the present paper offers nothing in this direction. As to "theoretical" results, the standard is a worst-case analysis. (A different question is average-case analysis, where the problem is to choose a convincing input distribution.) The most interesting and powerful results are those where all algorithmic techniques are allowed. A very basic goal is to achieve polynomial time.

One can then restrict to special types of algorithms by disallowing certain techniques. Usually this is done either for practical or for historical considerations:

1.a. Algorithms without restrictions,

1.b. algorithms that disallow:

α. some types of fast arithmetic,

β. probabilistic choice.

As an example of 1.b.α, it seems that "fast explicit" linear algebra, using $O(n^\omega)$ operations with $\omega < 3$, is not practical within current ranges, and so it may be quite reasonable to work with $O(n^3)$ "classical" linear algebra. In applications such as our factoring problem, both methods are no match for "fast implicit" probabilistic linear algebra à la Wiedemann, mainly because of memory requirements. For polynomial arithmetic, one can either use "classical" $O(n^2)$ or "fast" FFT-based $O(n \log n \operatorname{loglog} n)$ methods. The "classical" model is intuitive but has no rigorous definition. The popular belief that "fast" arithmetic is impractical was disproved, in a sense, by Shoup [**38**] who showed its superiority already for degree 25 modulo a 100-bit prime. A popular approach to killing two birds with one stone is by using some notation such as $\mathsf{M}(n)$ in Section 3, into which the reader can substitute the cost of her favorite polynomial arithmetic.

One is tempted to use a similar notation, say $\mathsf{L}(A)$, for the cost of linear algebra on a matrix $A \in F^{n \times n}$ given implicitly by a "black box". Thus if the

product of A with any vector in F^n can be computed with $\mathsf{P}(A)$ operations, then

$$\mathsf{L}(A) = \begin{cases} O(n\mathsf{P}(A) + n^3) & \text{with classical linear algebra,} \\ O(n\mathsf{P}(A) + n^\omega) & \text{with fast explicit linear algebra,} \\ O(n\mathsf{P}(A) + n^2 \log n \log\log n) & \text{with fast implicit linear algebra,} \end{cases}$$

where the first summands for the "classical" and "fast explicit" cases correspond to computing the n^2 entries of A by applying A to the unit vectors. Unfortunately, the last line of $\mathsf{L}(A)$ has not yet been validated, in that some problems, such as finding a basis for the nullspace, have not been solved by these methods, and there is the extra factor E from (3.1) for very small fields. Furthermore, this will not apply to matrix multiplication, but, at best, to problems like solving linear equations, computing characteristic or minimal polynomials, and maybe normal forms.

The restriction 1.b.β to deterministic algorithms comes from a long tradition. These algorithms are conceptually simpler than general algorithms (with probabilistic choice), and the model has a rigorous mathematical definition. We expand on the history of probabilistic algorithms below. There are several types of probabilistic algorithms; the ones for factoring polynomials over finite fields are all of the "Las Vegas" type and in practice as good as deterministic ones. However, the question of deterministic polynomial-time algorithms for factoring polynomials over finite fields of large characteristic remains a major theoretical challenge in the area. There are some results for special fields [**1, 13, 24, 26**] and for special polynomials [**17, 34, 35**], but the most spectacular recent progress is Evdokimov's algorithm [**9**] with time $(n^{\log n} \log q)^{O(1)}$, under the Extended Riemann Hypothesis. It is well recognized that even a complete solution of this challenge may very well not have any impact on practical algorithms.

What are the implications of Niederreiter's method—including the observations presented here—for practical computations? In response to the Polynomial Factorization Challenge (von zur Gathen [**14**]), Monagan [**27**] has shown that MAPLE can factor a polynomial of degree 200 modulo a prime with 200 bits in a "routine calculation", i.e., a day's time on a current workstation. Such a problem is given by about 40 000 bits of input, and it is reasonable to ask: can we factor routinely polynomials of degree 40 000 over $\mathbb{F}_2$? Of degree 100 000? Approaches using explicit linear algebra seem hopeless. Kaltofen [**19**] and Diaz *et al.*[**8**] report on solving sparse systems of 100 000 linear equations in $\mathbb{F}_2$ on a network of computers, using a Wiedemann approach. An "explicit" approach of storing such a matrix densely, with over 1 GByte, and operating on it, is infeasible in practice. Kaltofen & Lobo [**20**] have factored polynomials of degree 10 000 over $\mathbb{F}_{127}$, corresponding to about 70 000 input bits, in about four days on a network of eight Sun 4 workstations. Berlekamp's and Niederreiter's matrices have $\Omega(n^2)$ nonzero entries for most inputs f. It is not even clear how to use sparseness of f to speed up substantially an algorithm like Gaussian elimination. However, we consider it quite likely that in the not too far future we will routinely solve

gigantic factorization problems such as the one above.

A drawback of Kaltofen & Saunders' method [**21**] is that one has to work over a field with $\Omega(n^2 \log n)$ elements, thus requiring an extension of degree about $2 \log_2 n$ if one wants to factor over $\mathbb{F}_2$. We hope that this is just an artifact of our current state of knowledge, since Wiedemann's original algorithm—which does not solve our problem—and Coppersmith's variant work [**7**] also with random choices from $\mathbb{F}_2$. It would be interesting to see whether this is also possible for the problem at hand.

Both Berlekamp's (see [**12**]) and Niederreiter's method can be applied to polynomials that are not squarefree; Berlekamp's algorithm yields the primary factors, the maximal powers of irreducible polynomial divisors. It is not clear to us whether or not this is preferable to starting with the squarefree part of an arbitrary input polynomial. The latter is easily calculated, at not much more cost than a gcd (see [**42**]), and the more costly general algorithms will be applied to a smaller input. When there is a reason to suspect the presence of small factors, as is the case for random inputs, it may be interesting to extract these by an efficient distinct-degree procedure (as in [**15**]) that computes the gcd with $x^{q^i} - x$ for small i, say $i \leq \log n$ or $i \leq \sqrt{n}$. Then one has problems of two different types: small degree with many factors, and large degree with few factors. For the latter, say with r factors, the difference between r and $2 \log r$ might not be too large any more. This approach will not be fruitful when the input has special structure, as will usually be the case when it is produced by some resultant or norm computation.

Some parts of Niederreiter's algorithm work for arbitrary fields of positive characteristic [**28**]. We only discuss finite fields here. One reason is that over sufficiently general "computable" fields, without some restriction such as perfectness, even the question of squarefreeness is undecidable [**11**, 5.10].

We conclude with some philosophical remarks about probabilistic algorithms. Their introduction in the mid-70's amounted to a "scientific revolution" [**22**] in Computer Science, though Monte Carlo methods in numerical computing had been used since the 1940's. Berlekamp's (1970) algorithm for factoring polynomials over finite fields of large characteristic was the first instance where they solved a problem probabilistically in polynomial time that has no such deterministic solution (until today!). Unfortunately, the fundamental importance of Berlekamp's methodology was not generally recognized, and the technique only gained popular acceptance when Solovay & Strassen [**40**] applied it to the more intuitive problem of testing integers for primality. Today they are ubiquitous in algorithm design, where they are often either (exponentially) more efficient or much easier to program than deterministic algorithms. The probabilistic methods discussed here are of the "Las Vegas" type–genius loci–and return either "failure" or the correction answer. For probabilistically solved problems, it remains a theoretical challenge to see whether they can also be solved deterministically without too much loss in efficiency. Neither for implicit linear algebra nor for factoring in

large characteristic are such solutions in sight.

From a practical point of view, the fairest comparison of an algorithm's efficiency is to the best known algorithms, as in 1.a, for the problem at hand. (Besides methods based on linear algebra, the competition here are the algorithms of Cantor & Zassenhaus [**6**] and von zur Gathen & Shoup [**15**].) We have to wait for careful and comparable implementations to see which method performs best for which range of inputs. Several new ideas have emerged recently, and exciting times lie ahead for the polynomial factorizer.

Acknowledgement. Many thanks go to Erich Kaltofen and Harald Niederreiter for useful discussions on polynomial factorization, and to Jürgen Gerhard for correcting an error in one of our estimates.

References

1. E. Bach, J. von zur Gathen, and H. W. Lenstra Jr., *Deterministic factorization of polynomials over special finite fields*, Preprint, 1993.
2. E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Tech. J. **46** (1967), 1853–1859.
3. ———, *Factoring polynomials over large finite fields*, Math. Comp. **24** (1970), 713–735.
4. P. Camion, *A deterministic algorithm for factorizing polynomials of* $\mathbb{F}_q[x]$, Ann. Discr. Math. **17** (1983), 149–157.
5. D. G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta. Inform. **28** (1991), 693–701.
6. D. G. Cantor and H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. **36** (1981), 587–592.
7. D. Coppersmith, *Solving linear equations over* $GF(2)$*: Block Lanczos algorithm*, Lin. Alg. Appl. **192** (1993), 33–60.
8. A. Diaz, M. Hitz, E. Kaltofen, A. Lobo, and T. Valente, *Process scheduling in DCS and the large sparse linear systems challenge*, Proc. DISCO '93, Lect. Notes Comput. Sci, vol. 722, Springer Verlag, 1993, pp. 66–80.
9. S. A. Evdokimov, *Efficient factorization of polynomials over finite fields and the generalized Riemann hypothesis*, Technical Report, Universität Bonn, 1993.
10. P. Fleischmann, *Connections between the algorithms of Berlekamp and Niederreiter for factoring polynomials over* $\mathbb{F}_q$, Lin. Alg. Appl. **192** (1993), 101–108.
11. J. von zur Gathen, *Hensel and Newton methods in valuation rings*, Math. Comp. **42** (1984), 637–661.
12. ———, *Parallel algorithms for algebraic problems*, SIAM J. Comput. **13** (1984), 802–824.
13. ———, *Factoring polynomials and primitive elements for special primes*, Theoret. Computer Science **52** (1987), 77–89.
14. ———, *A polynomial factorization challenge*, SIGSAM Bulletin **26** (1992), 22–24.
15. J. von zur Gathen and V. Shoup, *Computing Frobenius maps and factoring polynomials*, Comput complexity **2** (1992), 187–224.
16. R. Göttfert, *An acceleration of the Niederreiter factorization algorithm in characteristic 2*, Math. Comp. (1994), To appear.
17. M.-D. A. Huang, *Riemann hypothesis and finding roots over finite fields*, Proc. 17th Ann. ACM Symp. Theory of Computing (Providence RI), 1985, pp. 121–130.
18. E. Kaltofen, *Polynomial factorization 1987–1991*, Proc. Latin'92, Lecture Notes in Computer Science **583** (São Paulo, Brazil), 1992, pp. 294–313.
19. ———, *Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems*, Tech. Report 93-22, Department of Computer Science, Rensselaer Polytechnic Institute, 1993.
20. E. Kaltofen and A. Lobo, *Factoring high-degree polynomials by the black box Berlekamp al-*

gorithm, Tech. report, Department of Computer Science, Rensselaer Polytechnic Institute, 1994.
21. E. Kaltofen and B. D. Saunders, *On Wiedemann's method of solving sparse linear systems*, Proc. AAECC-10, Springer Lecture Notes in Computer Science, 1991, pp. 29–38.
22. T. S. Kuhn, *The structure of scientific revolutions*, second ed., University of Chicago Press, Chicago IL, 1970.
23. T. C. Y. Lee and S. A. Vanstone, *Subspaces and polynomial factorizations over finite fields*, Preprint, 1992.
24. M. Mignotte and C. Schnorr, *Calcul des racines d-ièmes dans un corps fini*, C. R. Acad. Sci. Paris **290** (1988), 205–206.
25. V. S. Miller, *On the factorization method of Niederreiter*, Preprint, 1992.
26. R. T. Moenck, *On the efficiency of algorithms for polynomial factoring*, Math. Comp. **31** (1977), 235–250.
27. M. B. Monagan, *von zur Gathen's factorization challenge*, ACM SIGSAM Bull. **20** (1993), 13–18.
28. H. Niederreiter, *Factorization of polynomials and some linear-algebra problems over finite fields*, Lin. Alg. Appl. **192** (1993), 301–328.
29. ———, *New deterministic factorization algorithms for polynomials over finite fields*, Finite Fields: Theory, Applications and Algorithms (G. L. Mullen and P. J.-S. Shiue, eds.), Contemporary Mathematics, Amer. Math. Soc., 1994.
30. ———, *A new efficient factorization algorithm for polynomials over small finite fields*, Appl. Alg. Eng. Comm. Comp. **4** (1993), 81–87.
31. ———, *Factoring polynomials over finite fields using differential equations and normal bases*, Math. Comp., to appear, 1994.
32. H. Niederreiter and R. Göttfert, *Factorization of polynomials over finite fields and characteristic sequences*, Preprint, 1992.
33. ———, *On a new factorization algorithm for polynomials over finite fields*, Preprint, 1993.
34. L. Rónyai, *Factoring polynomials over finite fields*, J. Algorithms **9** (1988), 391–400.
35. ———, *Galois groups and factoring polynomials over finite fields*, SIAM J. Disc. Math. **5** (1992), 345–365.
36. A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing **7** (1971), 281–292.
37. V. Shoup, *On the deterministic complexity of factoring polynomials over finite fields*, Inform. Process. Lett. **33** (1990), 261–267.
38. ———, *Factoring polynomials over finite fields: asymptotic complexity vs. reality*, Proc. Int. IMACS Symp. on Symbolic Computation, New Trends and Developments (Lille, France), 1993, pp. 124–129.
39. I. E. Shparlinski, *Computational and algorithmic problems in finite fields*, Mathematics and its applications, vol. 88, Kluwer Academic Publishers, 1992.
40. R. Solovay and V. Strassen, *A fast Monte-Carlo test for primality*, SIAM J. Comput. **6** (1977), 84–85, Erratum **7** (1978), 118.
41. D. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inf. Theory **32** (1986), 54–62.
42. D. Y. Y. Yun, *On square-free decomposition algorithms*, Proc. ACM Symp. Symbolic and Algebraic Computation (R. D. Jenks, ed.), 1976, pp. 26–35.

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada

E-mail address: sgao@cs.toronto.edu and gathen@cs.toronto.edu